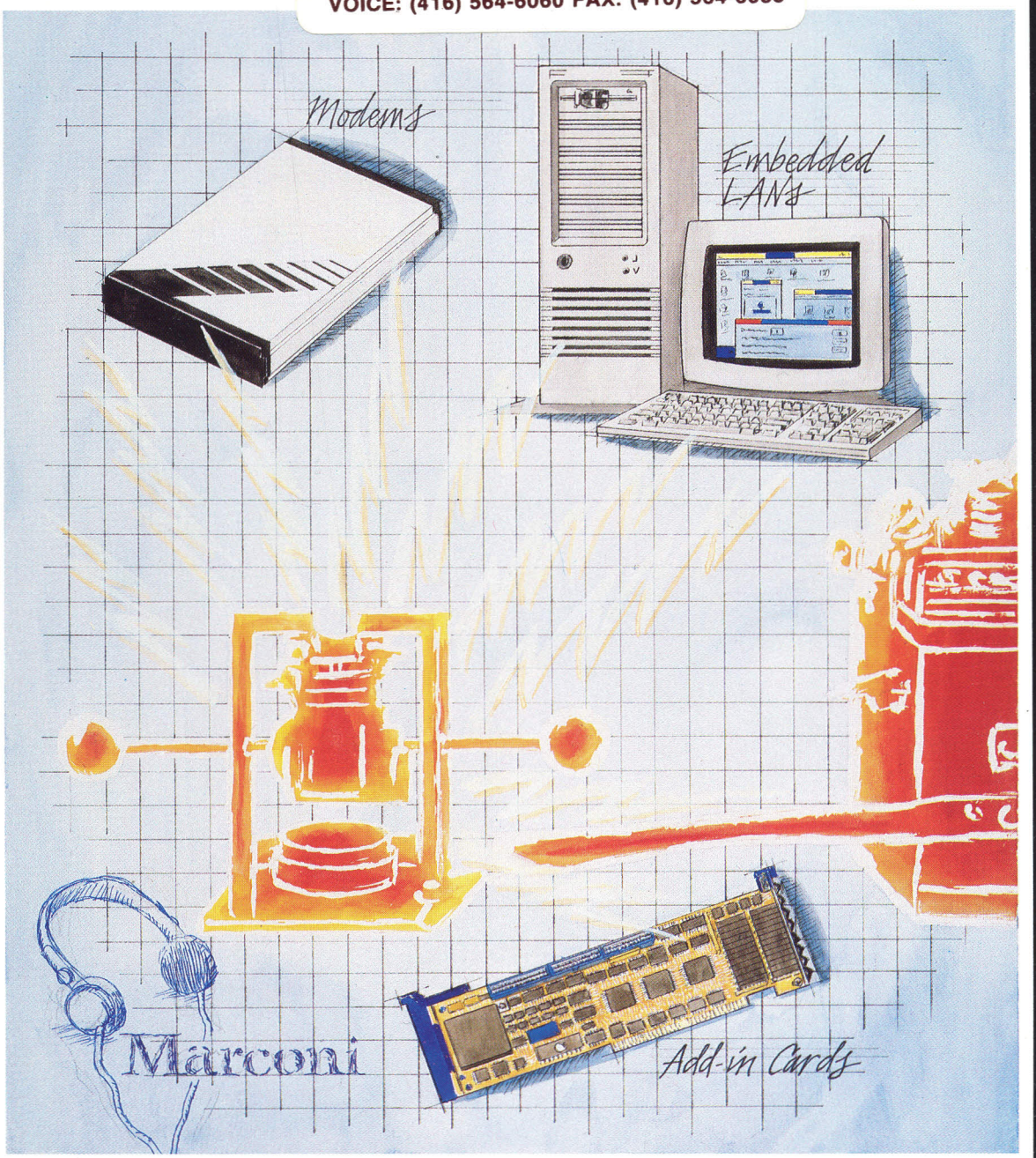
intel®

Microcommunications

HAMILTON/AVNET
151 SUPERIOR BLVD.
MISSISSAUGA, ONTARIO L5T 2L1, CANADA
VOICE: (416) 564-6060 FAX: (416) 564-6033



Benedict Norbert Wong

Order Number: 231658-006

intel®

LITERATURE

To order Intel Literature or obtain literature pricing information in the U.S. and Canada call or write Intel Literature Sales. In Europe and other international locations, please contact your ***local*** sales office or distributor.

INTEL LITERATURE SALES
P.O. BOX 7641
Mt. Prospect, IL 60056-7641

In the U.S. and Canada
call toll free
(800) 548-4725

CURRENT HANDBOOKS

Product line handbooks contain data sheets, application notes, article reprints and other design information. All handbooks can be ordered individually, and most are available in a pre-packaged set in the U.S. and Canada.

TITLE	INTEL ORDER NUMBER	ISBN
SET OF THIRTEEN HANDBOOKS (Available in U.S. and Canada)	**231003**	**N/A**
CONTENTS LISTED BELOW FOR INDIVIDUAL ORDERING:		
COMPONENTS QUALITY/RELIABILITY	210997	1-55512-132-2
EMBEDDED APPLICATIONS	270648	1-55512-123-3
8-BIT EMBEDDED CONTROLLERS	270645	1-55512-121-7
16-BIT EMBEDDED CONTROLLERS	270646	1-55512-120-9
16/32-BIT EMBEDDED PROCESSORS	270647	1-55512-122-5
MEMORY PRODUCTS	210830	1-55512-117-9
MICROCOMMUNICATIONS	231658	1-55512-119-5
MICROCOMPUTER PRODUCTS	280407	1-55512-118-7
MICROPROCESSORS	230843	1-55512-115-2
PACKAGING	240800	1-55512-128-4
PERIPHERAL COMPONENTS	296467	1-55512-127-6
PRODUCT GUIDE (Overview of Intel's complete product lines)	210846	1-55512-116-0
PROGRAMMABLE LOGIC	296083	1-55512-124-1
ADDITIONAL LITERATURE: (Not included in handbook set)		
AUTOMOTIVE HANDBOOK	231792	1-55512-125-x
INTERNATIONAL LITERATURE GUIDE (Available in Europe only)	E00029	N/A
CUSTOMER LITERATURE GUIDE	210620	N/A
MILITARY HANDBOOK (2 volume set)	210461	1-55512-126-8
SYSTEMS QUALITY/RELIABILITY	231762	1-55512-046-6

intel®

Microcommunications

Microcommunications is the application of microelectronic or silicon technology to the field of communications. Intel, long recognized as an established leader in the microcommunications market, has a broad range of components covering local area networks (LANs) and modems.

This handbook contains product data sheets, design and applications information for Intel's entire line of microcommunications products, including Intel's first 32-bit LAN coprocessor, the 82596; twisted-pair Ethernet* products; and modem chips for desktop and laptop PCs. Technical information on Intel's communications boards and supercomponents is also included. Application articles explore Intel's entire line of microcommunications products. Charts are included to show parameters and test conditions, layout principles and much more.

*Ethernet is a registered trademark of Xerox Corp.

Printed in USA/0191/35K/RRD/LK
Microcommunications

ISBN# 1-55512-119-5

Intel Corporation is a leading supplier of microcomputer components, modules and systems. When Intel invented the microprocessor in 1971, it created the era of the microcomputer. Today, Intel architectures are considered world standards. Whether used in embedded applications such as automobiles, printers and microwave ovens, or as the CPU in personal computers, client servers or supercomputers, Intel delivers leading-edge technology.

MICROCOMMUNICATIONS

1991

About Our Cover:

Thinkers, inventors, and artists throughout history have breathed life into their ideas by converting them into rough working sketches, models, and products. This series of covers shows a few of these creations, along with the applications and products created by Intel customers.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

287, 376, 386, 387, 486, 4-SITE, Above, ACE51, ACE96, ACE186, ACE196, ACE960, ActionMedia, BITBUS, COMMputer, CREDIT, Data Pipeline, DVI, ETOX, FaxBACK, Genius, i, i, i486, i750, i860, ICE, iCEL, ICEVIEW, iCS, iDBP, iDIS, I²ICE, iLBX, iMDDX, iMMX, Inboard, Insite, Intel, intel, Intel386, intelBOS, Intel Certified, Intelevision, intelligent Identifier, intelligent Programming, Intellec, Intellink, iOSP, iPAT, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, Pro750, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, READY LAN, RMX/80, RUPI, Seamless, SLD, SugarCube, ToolTALK, UPI, VAPI, Visual Edge, VLSiCEL, and ZapCode, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

CHMOS and HMOS are patented processes of Intel Corp.

PAL is a registered trademark of Advanced Micro Devices.

GAL is a registered trademark of Lattice Semiconductor, Inc.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

© INTEL CORPORATION 1990

intel®

CUSTOMER SUPPORT

INTEL'S COMPLETE SUPPORT SOLUTION WORLDWIDE

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, consulting services and network management services. For detailed information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is extensive. It can start with assistance during your development effort to network management. 100 Intel sales and service offices are located worldwide—in the U.S., Canada, Europe and the Far East. So wherever you're using Intel technology, our professional staff is within close reach.

HARDWARE SUPPORT SERVICES

Intel's hardware maintenance service, starting with complete on-site installation will boost your productivity from the start and keep you running at maximum efficiency. Support for system or board level products can be tailored to match your needs, from complete on-site repair and maintenance support to economical carry-in or mail-in factory service.

Intel can provide support service for not only Intel systems and emulators, but also support for equipment in your development lab or provide service on your product to your end-user/customer.

SOFTWARE SUPPORT SERVICES

Software products are supported by our Technical Information Service (TIPS) that has a special toll free number to provide you with direct, ready information on known, documented problems and deficiencies, as well as work-arounds, patches and other solutions.

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and; *COMMENTS Magazine*). Basic support consists of updates and the subscription service. Contracts are sold in environments which represent product groupings (e.g., iRMX® environment).

NETWORK SERVICE AND SUPPORT

Today's broad spectrum of powerful networking capabilities are only as good as the customer support provided by the vendor. Intel offers network services and support structured to meet a wide variety of end-user computing needs. From a ground up design of your network's physical and logical design to implementation, installation and network wide maintenance. From software products to turn-key system solutions; Intel offers the customer a complete networked solution. With over 10 years of network experience in both the commercial and Government arena; network products, services and support from Intel provide you the most optimized network offering in the industry.

CONSULTING SERVICES

Intel provides field system engineering consulting services for any phase of your development or application effort. You can use our system engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training and customizing an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

CUSTOMER TRAINING

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, BITBUS™ and LAN applications.

	the product becomes available.
Advanced Information	Contains information on products being sampled or in the initial production phase of development.*
Preliminary	Contains preliminary information on new products in production.*
No Marking	Contains information on products in full production.*

*Specifications within these data sheets are subject to change without notice. Verify with your local Intel sales office that you have the latest data sheet before finalizing a design.

Local Area Networks 1

Wide Area Networks 2

Other Components 3

Modem Products 4

PCM Codec/Filter and Combo 5

1 Local Area Networks

2 Wide Area Networks

3 Other Components

4 Modem Products

5 PCM Codec/Filter and Combo

82C501AD Ethernet Serial Interface 1-245
82521TB Twisted Pair Ethernet Serial Supercomponent 1-263
82504TA Transceiver Serial Interface (TSI) 1-283
82505TA Multiport Repeater Controller (MPR) 1-300
82506TB Twisted Pair Medium Attachment Unit (TP MAU) 1-321
APPLICATION NOTES
AP-235 An 82586 Data Link Driver 1-337
AP-236 Implementing StarLAN with the Intel 82588 1-418
AP-344 Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors 1-494
AP-345 Implementing 10BASE-T Networks with Intel's Twisted Pair Ethernet Components and Supercomponents 1-570
CSMA/CD Access Method Evaluation Tools
PC586E CSMA/CD LAN Evaluation Board 1-594

CHAPTER 2
Wide Area Networks
DATA SHEETS
8251A Programmable Communication Interface 2-1
82050 Asynchronous Communications Controller 2-26
82510 Asynchronous Serial Controller 2-40
8273 Programmable HDLC/SDLC Protocol Controller 2-80
8274 Multi-Protocol Serial Controller (MPSC) 2-112
82530/82530-6 Serial Communications Controller (SCC) 2-150
APPLICATION NOTES
AP-401 Designing With the 82510 Asynchronous Serial Controller 2-182
AP-310 High Performance Driver for 82510 2-263
AP-36 Using the 8273 SDLC/HDLC Protocol Controller 2-295
AP-134 Asynchronous Communication with the 8274 Multiple-Protocol Serial Controller 2-348
AP-145 Synchronous Communication with the 8274 Multiple Protocol Serial Controller 2-387
AP-222 Asynchronous and SDLC Communications with 82530 2-426

CHAPTER 3
Other Components
DATA SHEETS
8291A GPIB Talker/Listener 3-1
8292 GPIB Controller 3-33

Table of Contents (Continued)

8294A Data Encryption/Decryption Unit 3-52
APPLICATION NOTES
AP-166 Using the 8291A GPIB Talker/Listener 3-65
AP-66 Using the 8292 GPIB Controller 3-96

SECTION TWO—TELECOMMUNICATION COMPONENTS

CHAPTER 4
Modem Products
DATA SHEETS
89024 2400 BPS Intelligent Modem Chip Set 4-1
89C024LT Error Correcting Laptop Modem Chip-Set 4-23
89C024FT V.42/42bis Modem Chip Set 4-46
EVALUATION KITS
MD024LT Modem Demo Board 4-69
MD024FT Modem Demo Board 4-70
89024 MEK2 89024 Enhanced Modem Evaluation Kit 4-71
MEK3 Modem Evaluation Kit 4-73
APPLICATION BRIEF
AB-24 89024 Modem Customization for V.23 Data Transmission 4-75

CHAPTER 5
PCM Codec/Filter and Combo
DATA SHEETS
2910A PCM Codec-micro Law 8-Bit Companded A/D and D/A Converter 5-1
2911A-1 PCM Codec-A Law 8-Bit Companded A/D and D/A Converter 5-19
2912A PCM Transmit/Receive Filter 5-34
2913 and 2914 Combined Single-Chip PCM Codec and Filter 5-46
2916/2917 HMOS Combined Single Chip PCM Codec and Filter 5-67
APPLICATION NOTES
Applications Information 2910A/2911A/2912A 5-85
AP-142 Designing Second-Generation Digital Telephony Systems Using the Intel 2913/14 Codec/Filter Combochip 5-88

Alphanumeric Index

2910A PCM Codec-micro Law 8-Bit Companded A/D and D/A Converter 5-1
2911A-1 PCM Codec-A Law 8-Bit Companded A/D and D/A Converter 5-19
2912A PCM Transmit/Receive Filter 5-34
2913 and 2914 Combined Single-Chip PCM Codec and Filter 5-46
2916/2917 HMOS Combined Single Chip PCM Codec and Filter 5-67
82050 Asynchronous Communications Controller 2-26
82504TA Transceiver Serial Interface (TSI) 1-283
82505TA Multiport Repeater Controller (MPR) 1-300
82506TB Twisted Pair Medium Attachment Unit (TP MAU) 1-321
82510 Asynchronous Serial Controller 2-40
8251A Programmable Communication Interface 2-1
82521TB Twisted Pair Ethernet Serial Supercomponent 1-263
82530/82530-6 Serial Communications Controller (SCC) 2-150
82586 IEEE 802.3 Ethernet LAN Coprocessor 1-1
82588 High Integration Local Area Network Controller 1-186
82590 Advanced CSMA/CD LAN Controller with 8-Bit Data Path 1-212
82596CA High-Performance 32-Bit Local Area Network Coprocessor 1-38
82596DX and 82596SX High-Performance 32-Bit Local Area Network Coprocessor 1-110
8273 Programmable HDLC/SDLC Protocol Controller 2-80
8274 Multi-Protocol Serial Controller (MPSC) 2-112
8291A GPIB Talker/Listener 3-1
8292 GPIB Controller 3-33
8294A Data Encryption/Decryption Unit 3-52
82C501AD Ethernet Serial Interface 1-245
89024 2400 BPS Intelligent Modem Chip Set 4-1
89024 MEK2 89024 Enhanced Modem Evaluation Kit 4-71
89C024FT V.42/42bis Modem Chip Set 4-46
89C024LT Error Correcting Laptop Modem Chip-Set 4-23
AB-24 89024 Modem Customization for V.23 Data Transmission 4-75
AP-134 Asynchronous Communication with the 8274 Multiple-Protocol Serial Controller ... 2-348
AP-142 Designing Second-Generation Digital Telephony Systems Using the Intel 2913/14 Codec/Filter Combochip 5-88
AP-145 Synchronous Communication with the 8274 Multiple Protocol Serial Controller 2-387
AP-166 Using the 8291A GPIB Talker/Listener 3-65
AP-222 Asynchronous and SDLC Communications with 82530 2-426
AP-235 An 82586 Data Link Driver 1-337
AP-236 Implementing StarLAN with the Intel 82588 1-418
AP-302 Microcommunications Overview xii
AP-310 High Performance Driver for 82510 2-263
AP-344 Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors .. 1-494
AP-345 Implementing 10BASE-T Networks with Intel's Twisted Pair Ethernet Components and Supercomponents 1-570
AP-36 Using the 8273 SDLC/HDLC Protocol Controller 2-295
AP-401 Designing With the 82510 Asynchronous Serial Controller 2-182
AP-66 Using the 8292 GPIB Controller 3-96
Applications Information 2910A/2911A/2912A 5-85
MD024FT Modem Demo Board 4-70
MD024LT Modem Demo Board 4-69
MEK3 Modem Evaluation Kit 4-73
PC586E CSMA/CD LAN Evaluation Board 1-594

OVERVIEW

Imagine for a moment a world where all electronic communications were instantaneous. A world where voice, data, and graphics could all be transported via telephone lines to a variety of computers and receiving systems. A world where the touch of a finger could summon information ranging from stock reports to classical literature and bring it into environments as diverse as offices and labs, factories and living rooms.

Unfortunately, these promises of the Information Age still remain largely unfulfilled. While computer technology has accelerated rapidly over the last twenty years, the communications methods used to tie the wide variety of electronic systems in the world together have, by comparison, failed to keep pace. Faced with a tangle of proprietary offerings, high costs, evolving standards, and incomplete technologies, the world is still waiting for networks that are truly all-encompassing, the missing links to today's communications puzzle.

Enter microcommunications—microchip-based digital communications products and services. A migration of the key electronics communications functions into silicon is now taking place, providing the vital interfaces that have been lacking among the various networks now employed throughout the world. Through the evolution of VLSI (Very Large Scale Integration) technology, microcommunications now can offer the performance required to effect these communications interfaces at affordable costs, spanning the globe with silicon to eradicate the troublesome bottleneck that has plagued information transfer during recent years.

"There are three parts to the communications puzzle," says Gordon Moore, Intel Chairman and CEO. "The first incorporates the actual systems that communicate with each other, and the second is the physical means to connect them—such as cables, microwave technology, or fiber optics. It is the third area, the interfaces between the systems and the physical links, where silicon will act as the linchpin. That, in essence, is what microcommunications is all about."

THE COMMUNICATIONS BOTTLENECK

Visions of global networks are not new. Perhaps one of the most noteworthy of these has been espoused by Dr. Koji Kobayashi, chairman of NEC Corporation. His view of the future, developed over the nearly fifty years of his association with NEC, is known as C&C (Computers and Communications). It defines the marriage of passive communications systems and computers as processors and manipulators of information, providing the foundation for a discipline that is changing the basic character of modern society.

Kobayashi's macro vision hints at the obstacles confronting the future of C&C. When taken to the micro level, to silicon itself, one begins to understand the complexities that are involved. When Intel invented the microprocessor fifteen years ago, the first seeds of the personal computer revolution were sown , marking an era that over the last decade has dramatically influenced the way people work and live. PCs now proliferate in the office, in factories, and throughout laboratory environments. And their "intimidation" factor has lessened to where they are also becoming more and more prevalent in the home, beginning to penetrate a market that to date has remained relatively untapped.

Thanks to semiconductor technology, the personal computer has raised the level of productivity in our society. But most of that productivity has been gained by individuals at isolated workstations. Group productivity, meanwhile, still leaves much to be desired. The collective productivity of organizations can only be enhanced through more sophisticated networking technology. We are now faced with isolated "islands of automation" that must somehow be developed into networks of productivity.

But no amount of computing can meet these challenges if the corresponding communications technology is not sufficiently in step. The Information Age can only grow as fast as the lowest common denominator—which in this case is the aggregate communications bandwidth that continues to lag behind our increased computing power. Such is the nature of the communications bottleneck, where the growing amounts of information we are capable of generating can only flow as fast as the limited and incompatible communications capabilities now in place. Clearly, a crisis is at hand.

BREAKING UP THE BOTTLENECK

Three factors have contributed to this logjam: lack of industry standards, an insufficient cost/performance ratio, and the incomplete status of available communications technology to date.

- Standards—One look at the tangle of proprietary systems now populating office, factory, and laboratory environments gives a good indication of the inherent difficulty in hooking these diverse systems together. And these systems do not merely feature different architectures—they also represent completely different levels of computing, ranging from giant mainframes at one end of the scale down to individual microcontrollers on the other.

  The market has simply grown too fast to effectively accommodate the changes that have occurred. Suppliers face the dilemma of meshing product differentiation issues with industry-wide compatibility as

they develop their strategies; opting for one in the past often meant forsaking the other. And while some standards have coalesced, the industry still faces a technological Tower of Babel, with many proprietary solutions vying to be recognized in leadership positions.

- Cost/Performance Ratio—While various communications technologies struggle toward maturity, the industry has had to cope with tremendous costs associated with interconnectivity and interoperation. Before the shift to microelectronic interfaces began to occur, these connections often were prohibitively expensive.

  Says Ron Whittier, Intel Vice President and Director of Marketing: "Mainframes offer significant computing and communications power, but at a price that limits the number of users. What is needed is cost-effective communications solutions to hook together the roughly 16 million installed PCs in the market, as well as the soon-to-exist voice/data terminals. That's the role of microcommunications—bringing cost-effective communications solutions to the microcomputer world."

- Incomplete Technology—Different suppliers have developed many networking schemes, but virtually all have been fragmented and unable to meet the wide range of needs in the marketplace. Some of these approaches have only served to create additional problems, making OEMs and systems houses loathe to commit to suppliers who they fear cannot provide answers at all of the levels of communications that are now funneled into the bottleneck.

THE NETWORK TRINITY

Three principal types of networks now comprise the electronic communications marketplace: Wide Area Networks (WANs), Local Area Networks (LANs), and Small Area Networks (SANs). Each in its own fashion is turning to microcommunications for answers to its networking problems.

WANs—known by some as Global Area Networks (GANs)—are most commonly associated with the worldwide analog telephone system. The category also includes a number of other segments, such as satellite and microwave communications, traditional networks (like mainframe-to-mainframe connections), modems, statistical multiplexers, and front-end communications processors. The lion's share of nodes—electronic network connections—in the WAN arena, however, resides in the telecommunications segment. This is where the emerging ISDN (Integrated Services Digital Network) standard comes into focus as the most visible portion of the WAN marketplace.

The distances over which information may be transmitted via a WAN are essentially unlimited. The goal of ISDN is to take what is largely an analog global system and transform it into a digital network by defining the standard interfaces that will provide connections at each node.

These interfaces will allow basic digital communications to occur via the existing twisted pair of wires that comprise the telephone lines in place today. This would bypass the unfeasible alternative of installing completely new lines, which would be at cross purposes with the charter of ISDN: to reduce costs and boost performance through realization of an all-digital network.

The second category, Local Area Networks, represents the most talked-about link provided by microcommunications. In their most common form, LANs are comprised of—but not limited to—PC-to-PC connections. They incorporate information exchange over limited distances, usually not exceeding five kilometers, which often takes place within the same building or between adjacent work areas. The whole phenomenon surrounding LAN development, personal computing, and distributed processing essentially owes its existence to microcomputer technology, so it is not surprising that this segment of networking has garnered the attention it has in microelectronic circles.

Because of that, progress is being made in this area. The most prominent standard—which also applies to WANs and SANs—is the seven-layer Open Systems Interconnection (OSI) Model, established by the International Standards Organization (ISO). The model provides the foundation to which all LAN configurations must adhere if they hope to have any success in the marketplace. Interconnection protocols determining how systems are tied together are defined in the first five layers. Interoperation concepts are covered in the upper two layers, defining how systems can communicate with each other once they are tied together.

In the LAN marketplace, a large number of networking products and philosophies are available today, offering solutions at various price/performance points. Diverse approaches such as StarLAN, Token Bus and Token Ring, Ethernet, and PC-NET, to name a few of the more popular office LAN architectures, point to many choices for OEMs and end users.

A similar situation exists in the factory. While the Manufacturing Automation Protocol (MAP) standard is coalescing around the leadership of General Motors,

Boeing, and others, a variety of proprietary solutions also abound. The challenge is for a complete set of interfaces to emerge that can potentially tie all of these networks together in—and among—the office, factory, and lab environments.

The final third of the network trinity is the Small Area Network (SAN). This category is concerned with communications over very short distances, usually not exceeding 100 meters. SANs most often deal with chip-to-chip or chip-to-system transfer of information; they are optimized to deal with real-time applications generally managed by microcontrollers, such as those that take place on the factory floor among robots at various workstations.

SANs incorporate communications functions that are undertaken via serial backplanes in microelectronic equipment. While they represent a relatively small market in 1986 when compared to WANs and LANs, a tenfold increase is expected through 1990. SANs will have the greatest number of nodes among network applications by the next decade, thanks to their preponderance in many consumer products.

While factory applications will make up a large part of the SAN marketplace probably the greatest contributor to growth will be in automotive applications. Microcontrollers are now used in many dashboards to control a variety of engine tasks electronically, but they do not yet work together in organized and efficient networks. As Intel's Gordon Moore commented earlier this year to the New York Society of Security Analysts, when this technology shifts into full gear during the next decade, the total automobile electronics market will be larger than the entire semiconductor market was in 1985.

MARKET OPPORTUNITIES

Such growth is also mirrored in the projections for the WAN and LAN segments, which, when combined with SANs, make up the microcommunications market pie. According to Intel analysts, the total silicon microcommunications market in 1985 amounted to $522 million. By 1989, Intel predicts this figure will have expanded to $1290 million, representing a compounded annual growth rate of 25%.

And although the WAN market will continue to grow at a comfortable rate, the SAN and LAN pieces of the pie will increase the most dramatically. Whereas SANs represented only about 12.5% ($65 million) in 1985, they could explode to 22.5% ($290 million) of the larger pie by 1989. This growth is paralleled by increases in the LAN segment, which should grow from 34.5% of the total silicon microcommunications market in 1985 to 44.5% of the expanded pie in 1989.

Opportunities abound for microcommunications suppliers as the migration to silicon continues. And perhaps no VLSI supplier is as well-positioned in this marketplace as Intel, which predicts that 50% of its products will be microcommunications-related by 1990. The key here is the corporation's ability to bridge the three issues that contribute to the communications bottleneck: standards, cost-performance considerations, and the completeness of microcomputer and microcommunications product offerings.

INTEL AND VLSI: THE MICROCOMMUNICATIONS MATCH

Intel innovations helped make the microcomputer revolution possible. Such industry "firsts" include the microprocessor, the EPROM, the E2PROM, the microcontroller, development systems, and single board computers. Given this legacy, it is not surprising that the corporation should come to the microcommunications marketplace already equipped with a potent arsenal of tools and capabilities.

The first area centers on industry standards. As a VLSI microelectronic leader, Intel has been responsible for driving many of the standards that are accepted by the industry today. And when not actually initiating these standards, Intel has supported other existing and emerging standards through its longtime "open systems" philosophy. This approach protects substantial customer investments and ensures easy upgradability by observing compatibility with previous architectures and industry-leading standards.

Such a position is accentuated by Intel's technology relationships and alliances with many significant names in the microcommunications field. Giants like AT&T in the ISDN arena, General Motors in factory networking, and IBM in office automation all are working closely with Intel to further the standardization of the communications interfaces that are so vital to the world's networking future.

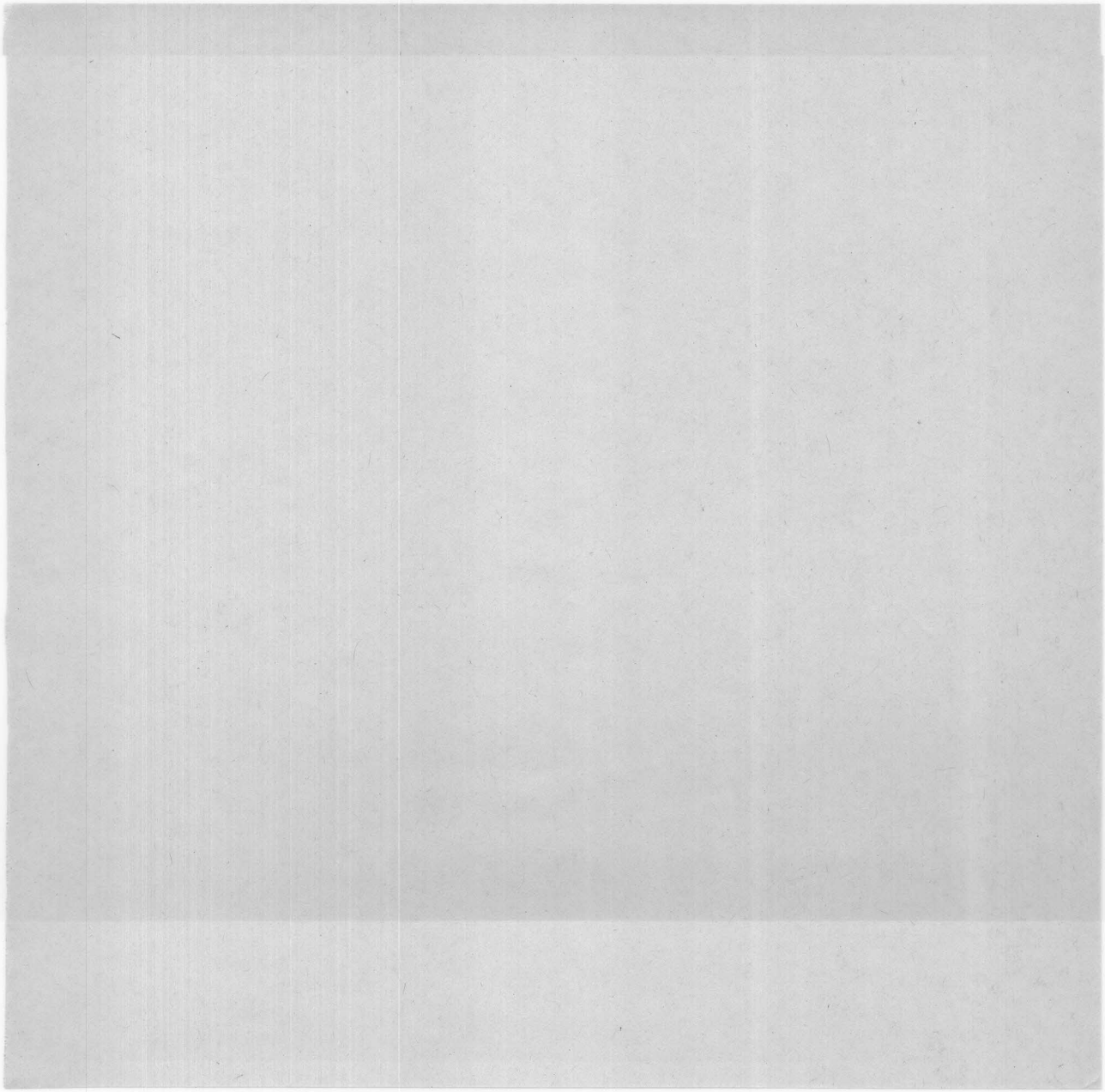
Cost/performance considerations also point to Intel's strengths. As a pioneer in VLSI technology, Intel has been at the forefront of achieving greater circuit densities and performance on single pieces of silicon: witness the 275,000 transistors housed on the 32-bit 80386, the highest performance commercial microprocessor ever built. As integration has increased, cost-per-bit has decreased steadily, marking a trend that remains consistent in the semiconductor industry. And one thing is

certain: microcommunications has a healthy appetite for transistors, placing it squarely in the center of the VLSI explosion.

But it is in the final area—completeness of technology and products—where Intel is perhaps the strongest. No other microelectronic vendor can point to as wide an array of products positioned across the various segments that comprise the microelectronic marketplace. Whether it be leadership in the WAN marketplace as the number one supplier of merchant telecommunications components, strength in SANs with world leadership in microcontrollers, or overall presence in the LAN arena with complete solutions in components, boards, software, and systems, Intel is a vital presence in the growing microcommunications arena.

That leadership extends beyond products. Along with its own application software, Intel is promoting expansion through partnerships with many different independent software vendors (ISVs), ensuring that the necessary application programs will be in place to fuel the gains provided by the silicon "engines" residing at the interface level. And finally, the corporation's commitment to technical support training, service, and its strong force of field applications engineers guarantees that it will back up its position and serve the needs that will continue to spring up as the microcommunications evolution becomes a reality.

Together, all the market segment alluded to in this article comprise the world of microcommunications, a world coming closer together every day as the web of networking solutions expands—all thanks to the technological ties that bind, reaching out to span the globe with silicon.

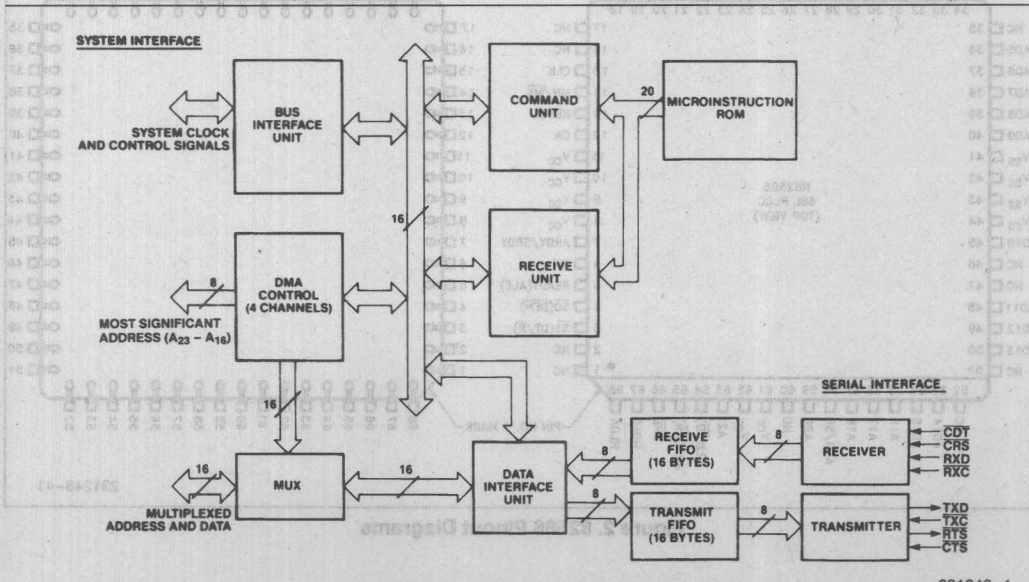


1 Local Area Networks

82586
IEEE 802.3 ETHERNET LAN COPROCESSOR

- **Performs Complete CSMA/CD Medium Access Control Functions Independently of CPU**
  - **High-Level Command Interface**
- **Supports Established and Emerging LAN Standards**
  - **IEEE 802.3/Ethernet (10BASE5)**
  - **IEEE 802.3/Cheapernet (10BASE2)**
  - **IEEE 802.3/StarLAN (1BASE5)**
  - **Proposed 10BASE-T**
  - **Proposed 10BASE-F**
  - **Proprietary CSMA/CD Networks up to 10 Mb/s**
- **On-Chip Memory Management**
  - **Automatic Buffer Chaining**
  - **Buffer Reclaim After Receipt of Bad Frames**
  - **Save Bad Frames, Optionally**
- **Interfaces to 8-Bit and 16-Bit Microprocessors**
- **48-Pin DIP and 68-Pin PLCC**
  (see "Intel Packaging" Document, Order Number: 231369-001)
- **Supports Minimum Component Systems**
  - **Shared Bus Configuration**
  - **Interface to 80186 and 80188 Microprocessors Without Glue**
- **Supports High-Performance Systems**
  - **Bus Master, with On-Chip DMA**
  - **5-MB/s Bus Bandwidth**
  - **Compatible with Dual-Port Memory**
  - **Back-to-Back Frame Reception at 10 Mb/s**
- **Network Management**
  - **CRC Error Tally**
  - **Alignment Error Tally**
  - **Location of Cable Faults**
- **Self-Test Diagnostics**
  - **Internal Loopback**
  - **External Loopback**
  - **Internal Register Dump**
  - **Backoff Timer Check**

1

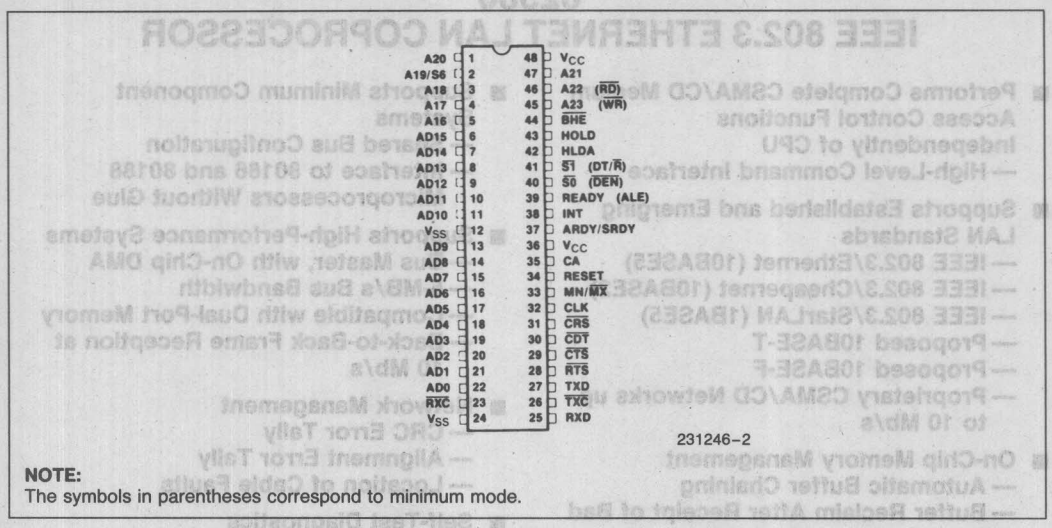


231246-1

Figure 1. 82586 Functional Block Diagram

*IBM is a trademark of International Business Machines Corporation.

September 1990
Order Number: 231246-007



NOTE:
The symbols in parentheses correspond to minimum mode.

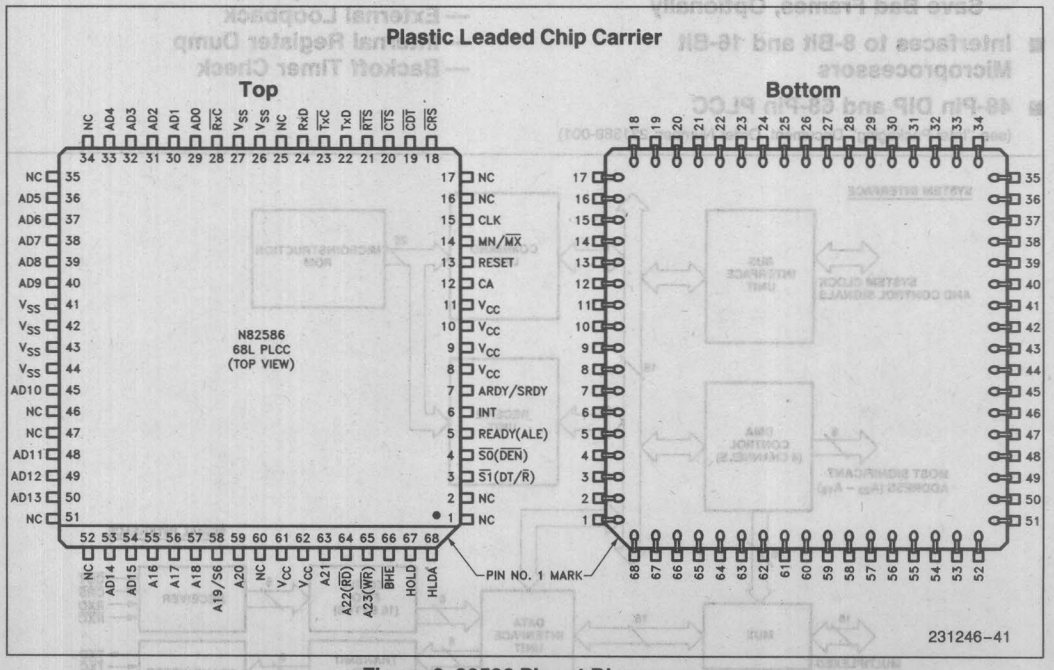


Figure 2. 82586 Pinout Diagrams

The 82586 is an intelligent, high-performance Local Area Network coprocessor, implementing the CSMA/CD access method (Carrier Sense Multiple Access with Collision Detection). It performs all time-critical functions independently of the host processor, which maximizes performance and network efficiency.

The 82586 performs the full set of IEEE 802.3 CSMA/CD Medium Access Control and channel interface functions including: framing, preamble generation and stripping, source address generation, destination address checking, CRC generation and checking, short frame detection. Any data rate up to 10 Mb/s can be used.

The 82586 features a powerful host system interface. It automatically manages memory structures with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels transparently to the user. Buffers containing errored or collided frames can be automatically recovered. The 82586 can be configured for 8-bit or 16-bit data path, with maximum burst transfer rate of 2 or 4 MB/s respectively. Memory address space is 16 megabytes maximum.

The 82586 provides two independent 16-byte FIFOs, one for receiving and one for transmitting. The threshold for block transfer to/from memory is programmable, enabling the user to optimize bus overhead for a given worst case bus latency.

The 82586 provides a rich set of diagnostic and network management functions including: internal and external loopbacks, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address, optional capture of errored or collided frames, and time domain reflectometry for locating faults in the cable.

The 82586 can be used in either baseband or broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) for any length network operating at any data rate up to 10 Mb/s. The controller supports address field lengths of 1, 2, 3, 4, 5, or 6 bytes. It can be configured for either the IEEE 802.3/Ethernet or HDLC method of frame delineation. Both 16-bit and 32-bit CRCs are supported.

The 82586 is fabricated in Intel's reliable HMOS II 5-V technology and is available in a 48-pin DIP or 68-pin PLCC package.

Table 1. 82586 Pin Description

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
V_{CC}, V_{CC}	48, 36	8, 9, 10, 11, 61, 62		System Power: +5V Power Supply.
V_{SS}, V_{SS}	12, 24	26, 27, 41, 42, 43, 44		System Ground.
RESET	34	13	I TTL	RESET is an active HIGH internally synchronized signal, causing the 82586 to terminate present activity immediately. The signal must be HIGH for at least four clock cycles. The 82586 will execute RESET within ten system clock cycles starting from RESET HIGH. When RESET returns LOW, the 82586 waits for the first CA to begin the initialization sequence.
TxD	27	22	0 TTL	Transmitted Serial Data output signal. This signal is HIGH when not transmitting.
$\overline{TxC}$	26	23	I *	Transmit Data Clock. This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ mode of operation, data is transferred to the TxD pin on the HIGH to LOW clock transition.
RxD	25	24	I TTL	Received Data Input Signal.
$\overline{RxC}$	23	28	I *	Received Data Clock. This signal provides timing information to the internal shifting logic depending upon the mode of data transfer. For NRZ data, the state of the RxD pin is sampled on the HIGH to LOW clock transition.

*See D.C. Characteristics.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
$\overline{RTS}$	28	21	O TTL	Request To Send signal. When LOW, notifies an external interface that the 82586 has data to transmit. It is forced HIGH after a Reset and while the Transmit Serial Unit is not sending data.
$\overline{CTS}$	29	20	I TTL	Active LOW Clear To Send input enables the 82586 transmitter to actually send data. It is normally used as an interface handshake to $\overline{RTS}$. This signal going inactive stops transmission. It is internally synchronized. If $\overline{CTS}$ goes inactive, meeting the setup time to $\overline{TxC}$ negative edge, transmission is stopped and $\overline{RTS}$ goes inactive within, at most, two TxC cycles.
$\overline{CRS}$	31	18	I TTL	Active LOW Carrier Sense input used to notify the 82586 that there is traffic on the serial link. It is used only if the 82586 is configured for external Carrier Sense. When so configured, external circuitry is required for detecting serial link traffic. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles.
$\overline{CDT}$	30	19	I TTL	Active LOW Collision Detect input is used to notify the 82586 that a collision has occurred. It is used only if the 82586 is configured for external Collision Detect. External circuitry is required for detecting the collision. It is internally synchronized. To be accepted, the signal must stay active for at least two serial clock cycles. During transmission, the 82586 is able to recognize a collision one bit time after preamble transmission has begun.
INT	38	6	O TTL	Active HIGH Interrupt request signal.
CLK	32	15	I MOS	The system clock input from the 80186 or another symmetrical clock generator.
MN/$\overline{MX}$	33	14	I TTL	When HIGH, MN/$\overline{MX}$ selects $\overline{RD}$, $\overline{WR}$, ALE $\overline{DEN}$, DT/$\overline{R}$ (Minimum Mode). When LOW, MN/$\overline{MX}$ selects A22, A23, READY, $\overline{S0}$, $\overline{S1}$ (Maximum Mode). Note: This pin should be static during 82586 operation.
AD0–AD15	6–11, 13–22	29–33, 36–40, 45, 48, 49, 50, 53, 54	I/O TTL	These lines form the time multiplexed memory address (t1) and data (t2, t3, tW, t4) bus. When operating with an 8-bit bus, the high byte will output the address only during T1. AD0–AD15 are floated after a RESET or when the bus is not acquired.
A16–A18 A20–A23	1, 3–5 45–47	55–57, 59, 63–65	O TTL	These lines constitute 7 out of 8 most significant address bits for memory operation. They switch during t1 and stay valid during the entire memory cycle. The lines are floated after RESET or when the bus is not acquired. Address lines A22 and A23 are not available for use in minimum mode.
A19/S6	2	58	O TTL	During t1 it forms line 19 of the memory address. During t2 through t4 it is used as a status indicating that this is a Master peripheral cycle, and is HIGH. Its timing is identical to that of AD0–AD15 during write operation.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
HOLD	43	67	O TTL	HOLD is an active HIGH signal used by the 82586 to request local bus mastership at the end of the current CPU bus transfer cycle, or at the end of the current DMA burst transfer cycle. In normal operation, HOLD goes inactive before HLDA. The 82586 can be forced off the bus by HLDA going inactive. In this case, HOLD goes inactive within four clock cycles in word mode and eight clock cycles in byte mode.
HLDA	42	68	I TTL	HLDA is an active HIGH Hold Acknowledge signal indicating that the CPU has received the HOLD request and that bus control has been relinquished to the 82586. It is internally synchronized. After HOLD is detected as LOW, the processor drives HLDA LOW. Note, CONNECTING V_{CC} TO HLDA IS NOT ALLOWED because it will cause a deadlock. Users wanting to give permanent bus access to the 82586 should connect HLDA with HOLD.
CA	35	12	I TTL	The CA pin is a Channel Attention input used by the CPU to initiate the 82586 execution of memory resident Command Blocks. The CA signal is synchronized internally. The signal must be HIGH for at least one system clock period. It is latched internally on HIGH to LOW edge and then detected by the 82586.
$\overline{BHE}$	44	66	O TTL	The Bus High Enable signal ($\overline{BHE}$) is used to enable data onto the most significant half of the data bus. Its timing is identical to that of A16–A23. With a 16-bit bus it is LOW and with an 8-bit bus it is HIGH. Note: after RESET, the 82586 is configured to 8-bit bus.
READY	39	5	I TTL	This active HIGH signal is the acknowledgement from the addressed memory that the transfer cycle can be completed. While LOW, it causes wait states to be inserted. This signal must be externally synchronized with the system clock. The Ready signal internal to the 82586 is a logical OR between READY and SRDY/ARDY.
ARDY/SRDY	37	7	I TTL	This active HIGH signal performs the same function as READY. If it is programmed at configure time to SRDY, it is identical to READY. If it is programmed to ARDY, the positive edge of the Ready signal is internally synchronized. Note, the negative edge must still meet setup and hold time specifications, when in ARDY mode. The ARDY signal must be active for at least one system clock HIGH period for proper strobing. The Ready signal internal to the 82586 is a logical OR between READY (in Maximum Mode only) and SRDY/ARDY. Note that following RESET, this pin assumes ARDY mode.

Table 1. 82586 Pin Description (Continued)

Symbol	48 Pin DIP Pin No.	68 Pin PLCC Pin No.	Type Level	Name and Function
$\overline{S0}$, $\overline{S1}$	40,41	4, 3	0 TTL	Maximum mode only. These status pins define the type of DMA transfer during the current memory cycle. They are encoded as follows: $\overline{S1}$ $\overline{S0}$ 0 0 Not Used 0 1 Read Memory 1 0 Write Memory 1 1 Passive Status is active from the middle of t4 to the end of t2. They return to the passive state during t3 or during tW when READY or ARDY is HIGH. These signals can be used by the 8288 Bus Controller to generate all memory control and timing signals.* Any change from the passive state, signals the 8288 to start the next t1 to t4 bus cycle. These pins are pulled HIGH and floated after a system RESET and when the bus is not acquired.
$\overline{RD}$	46	64	0 TTL	Used in minimum mode only. The read strobe indicates that the 82586 is performing a memory read cycle. $\overline{RD}$ is active LOW during t2, t3 and tW of any read cycle. This signal is pulled HIGH and floated after a RESET and when the bus is not acquired.
$\overline{WR}$	45	65	0 TTL	Used in minimum mode only. The write strobe indicates that the 82586 is performing a write memory cycle. $\overline{WR}$ is active LOW during t2, t3 and tW of any write cycle. It is pulled HIGH and floats after RESET and when the bus is not acquired.
ALE	39	5	0 TTL	Used in minimum mode only. Address Latch Enable is provided by the 82586 to latch the address into the 8282/8283 address latch. It is a HIGH pulse, during t1 ('clock low') of any bus cycle. Note that ALE is never floated.
$\overline{DEN}$	40	4	0 TTL	Used in minimum mode only. Data ENable is provided as output enable for the 8286/8287 transceivers in a stand-alone (no 8288) system. $\overline{DEN}$ is active LOW during each memory access. For a read cycle, it is active from the middle of t2 until the beginning of t4. For a write cycle, it is active from the beginning of t2 until the middle of t4. It is pulled HIGH and floats after a system RESET or when the bus is not acquired.
DT/$\overline{R}$	41	3	0 TTL	Used in minimum mode only. DT/$\overline{R}$ is used in non-8288 systems using an 8286/8287 data bus transceiver. It controls the direction of data flow through the Transceiver. Logically, DT/$\overline{R}$ is equivalent to $\overline{S1}$. It becomes valid in the t4 preceding a bus cycle and remains valid until the final t4 of the cycle. This signal is pulled HIGH and floated after a RESET or when the bus is not acquired.

NOTE:
*8288 does not support 10 MHz operation.

82586/HOST CPU INTERACTION

Communication between the 82586 and the host is carried out via shared memory. The 82586's on-chip DMA capability allows autonomous transfer of data blocks (buffers, frames) and relieves the CPU of byte transfer overhead. The 82586 is optimized to interface the iAPX 186, but due to the small number of hardware signals between the 82586 and the CPU, the 82586 can operate easily with other processors. The 82586/host interaction is explained separately in terms of the logical interface and the hardware bus interface.

The 82586 consists of two independent units: Command Unit (CU) and Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The CU and RU enable the 82586 to engage in the two types of activities simultaneously: the CU may be fetching and executing commands out of memory, and the RU may be storing received frames in memory. CPU intervention is only required after the CU executes a sequence of commands or the RU stores a sequence of frames.

The only hardware signals that connect the CPU and the 82586 are INTERRUPT and CHANNEL ATTENTION (see Figure 3). Interrupt is used by the 82586 to draw the CPU's attention to a change in the contents of the SCB. Channel Attention is used by the CPU to draw the 82586's attention.

82586 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: Initialization Root, System Control Block (SCB), Command List, and Receive Frame Area (RFA) (see Figure 4).

The Initialization Root is at a predetermined location in the memory space, (0FFFFF6H), known to both the host CPU and the 82586. The root is accessed at initialization and points to the System Control Block.

The System Control Block (SCB) functions as a bidirectional mail drop between the host CPU, CU and RU. It is the central element through which the CPU and the 82586 exchange control and status information. The SCB consists of two parts, the first of which entails instructions from the CPU to the 82586. These include: control of the CU and RU (START, ABORT, SUSPEND, RESUME), a pointer to the list of commands for the CU, a pointer to the receive frame area, and a set of Interrupt acknowledge bits. The second entails status information keyed by the 82586 to the CPU, including: state of the CU and RU (e.g. IDLE, ACTIVE READY, SUSPENDED, NO RECEIVE RESOURCES), interrupts bits (command completed, frame received, CU not ready, RU not ready), and statistics (see Figure 4).

The Command List serves as a program for the CU. Individual commands are placed in memory units called a Command Block, or CB. CB's contain command specific parameters and command specific statuses. Specifically, these high level commands are called Action Commands (e.g. Transmit, Configure).

A specific command, Transmit, causes transmission of a frame by the 82586. The Transmit command block includes Destination Address, Length Field, and a pointer to a list of linked buffers that holds the frame to be constructed from several buffers scattered in memory. The Command Unit performs with-

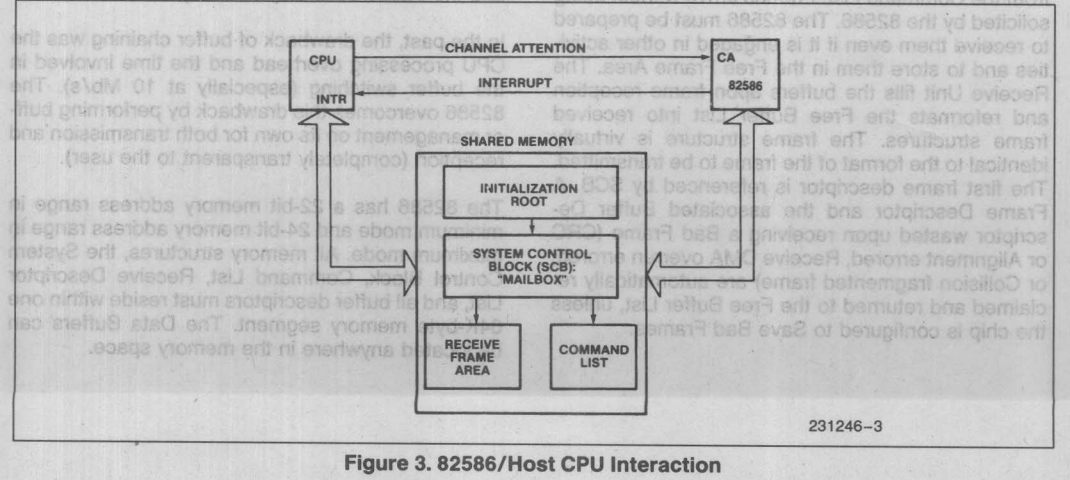


Figure 3. 82586/Host CPU Interaction

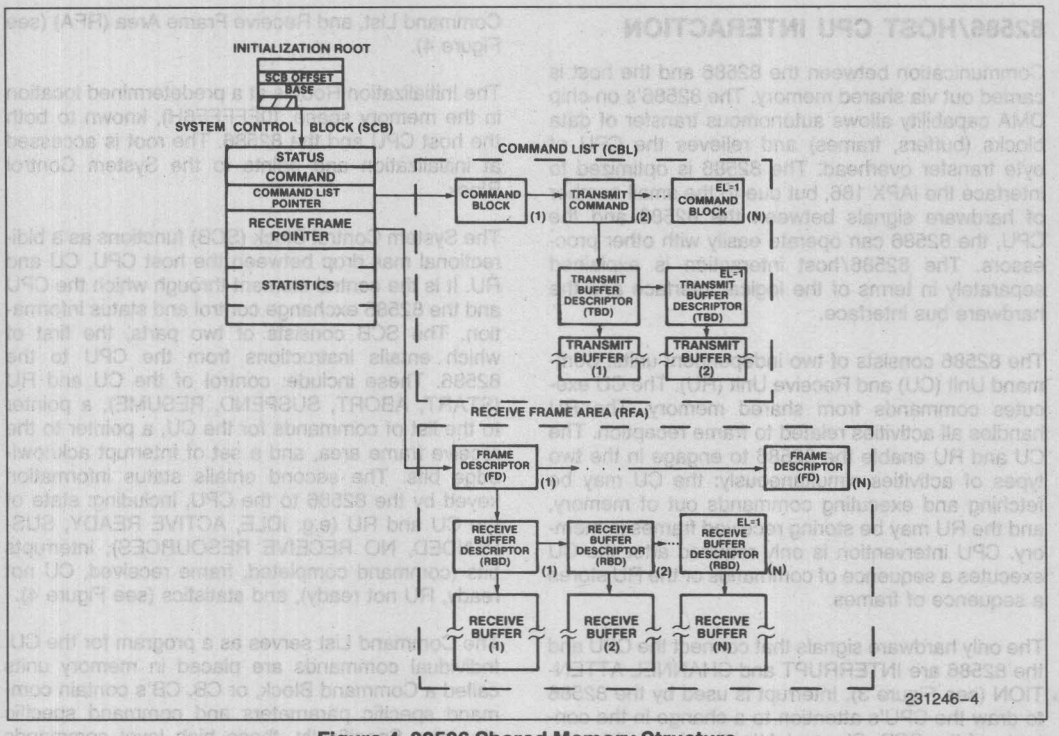


Figure 4. 82586 Shared Memory Structure

out the CPU intervention, the DMA of each buffer and the prefetching of references to new buffers in parallel. The CPU is notified only after successful transmission or retransmission.

The Receive Frame Area is a list of Free Frame Descriptors (Descriptors not yet used) and a list of buffers prepared by the user. It is conceptually distinct from the Command List. Frames arrive without being solicited by the 82586. The 82586 must be prepared to receive them even if it is engaged in other activities and to store them in the Free Frame Area. The Receive Unit fills the buffers upon frame reception and reformats the Free Buffer List into received frame structures. The frame structure is virtually identical to the format of the frame to be transmitted. The first frame descriptor is referenced by SCB. A Frame Descriptor and the associated Buffer Descriptor wasted upon receiving a Bad Frame (CRC or Alignment errored, Receive DMA overrun errored, or Collision fragmented frame) are automatically reclaimed and returned to the Free Buffer List, unless the chip is configured to Save Bad Frames.

Receive buffer chaining (i.e. storing incoming frames in a linked list of buffers) improves memory utilization significantly. Without buffer chaining, the user must allocate consecutive blocks of the maximum frame size (1518 bytes in Ethernet) for each frame. Taking into account that a typical frame size may be about 100 bytes, this practice is very inefficient. With buffer chaining, the user can allocate small buffers and the 82586 uses only as many as needed.

In the past, the drawback of buffer chaining was the CPU processing overhead and the time involved in the buffer switching (especially at 10 Mb/s). The 82586 overcomes this drawback by performing buffer management on its own for both transmission and reception (completely transparent to the user).

The 82586 has a 22-bit memory address range in minimum mode and 24-bit memory address range in maximum mode. All memory structures, the System Control Block, Command List, Receive Descriptor List, and all buffer descriptors must reside within one 64K-byte memory segment. The Data Buffers can be located anywhere in the memory space.

TRANSMITTING FRAMES

The 82586 executes high level action commands from the Command List in external memory. Action commands are fetched and executed in parallel with the host CPU's operation, thereby significantly improving system performance. The general action commands format is shown in Figure 5.

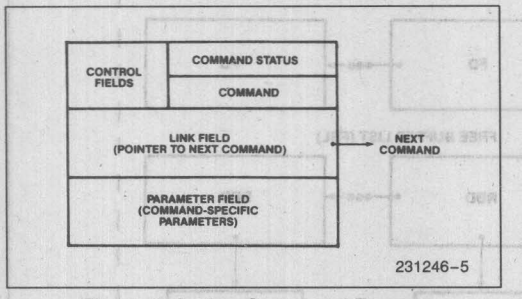


Figure 5. Action Command Format

Message transmission is accomplished by using the Transmit command. A single Transmit command contains, as part of the command-specific parameters, the destination address and length field for the transmitted frame along with a pointer to a buffer area in memory containing the data portion of the frame. (See Figure 15.) The data field is contained in a memory data structure consisting of a Buffer Descriptor (BD) and Data Buffer (or a linked list of buffer descriptors and buffers) as shown in Figure 6. The BD contains a Link Field which points to the next BD on the list and a 24-bit address pointing to the Data Buffer itself. The length of the Data Buffer is specified by the Actual Count field of the BD.

Using the BD's and Data Buffers, multiple Data Buffers can be 'chained' together. Thus, a frame with a long Data Field can be transmitted using multiple (shorter) Data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management policies.

The 82586 automatically generates the preamble (alternating 1's and 0's) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field from

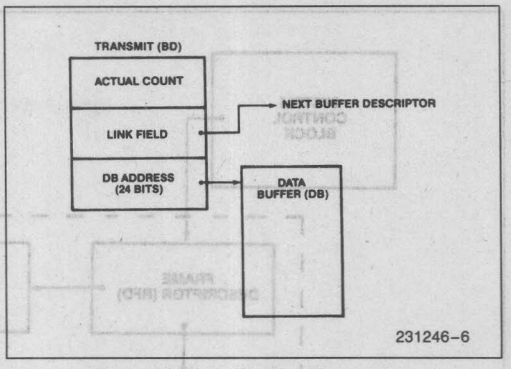


Figure 6. Data Buffer Descriptor and Data Buffer Structure

buffers pointed to by the Transmit command, and computes and appends the CRC at the end of the frame. See Figure 7.

The 82586 can be configured to generate either the Ethernet or HDLC start and end frame delimiters. In the Ethernet mode, the start frame delimiter is 10101011 and the end frame delimiter indicated by the lack of a signal after transmitting the last bit of the frame check sequence field. When in the HDLC mode, the 82586 will generate the 01111110 'flag' for the start and end frame delimiters and perform the standard 'bit stuffing/stripping'. In addition, the 82586 will optionally pad frames that are shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

In the event of a collision (or collisions), the 82586 manages the entire jam, random wait and retry process, reinitializing DMA pointers without CPU intervention. Multiple frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message that is larger than the maximum frame size (1518 bytes for Ethernet).

RECEIVING FRAMES

In order to minimize CPU overhead, the 82586 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate

PREAMBLE	START FRAME DELIMITER	DEST ADDR	SOURCE ADDR	LENGTH FIELD	DATA FIELD	FRAME CHECK SEQUENCE	END FRAME DELIMITER

Figure 7. Frame Format

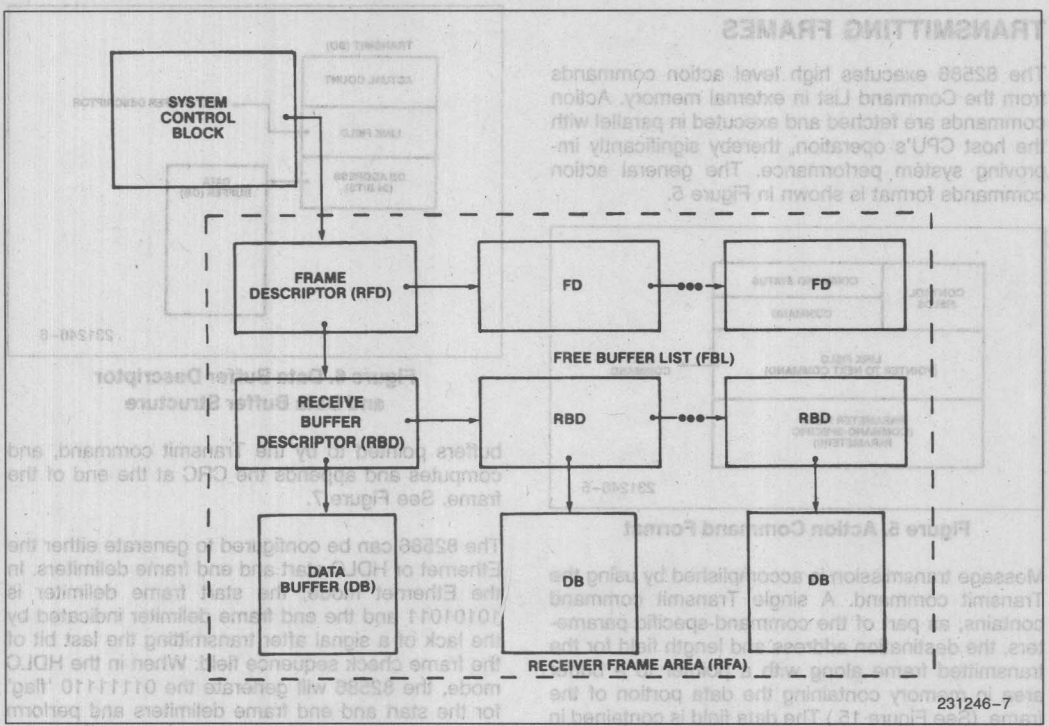


Figure 8. Receive Frame Area Diagram

amount of receive buffer space and then enables the 82586's Receive Unit. Once enabled, the RU 'watches' for any of its frames which it automatically stores in the Receive Frame Area (RFA). The RFA consists of a Receive Descriptor List (RDL) and a list of free buffers called the Free Buffer List (FBL) as shown in Figure 8. The individual Receive Frame Descriptors that make up the RDL are used by the 82586 to store the destination and source address, length field and status of each frame that is received. (Figure 9.)

The 82586, once enabled, checks each passing frame for an address match. The 82586 will recognize its own unique address, one or more multicast addresses or the broadcast address. If a match occurs, it stores the destination and source address and length field in the next available RFD. It then begins filling the next free Data Buffer on the FBL (which is pointed to by the current RFD) with the data portion of the incoming frame. As one DB is filled, the 82586 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers that fit a frame size that may be much shorter than the maximum allowable frame.

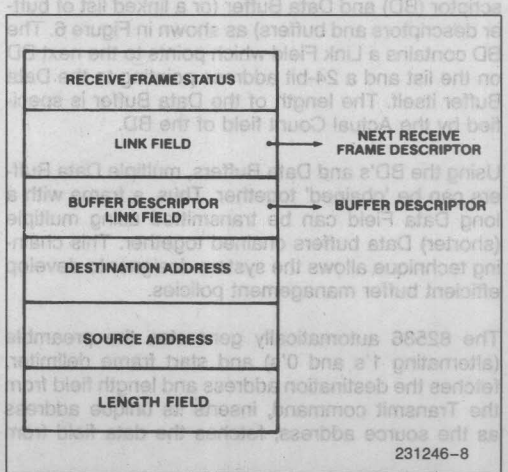


Figure 9. Receive Frame Descriptor

Once the entire frame is received without error, the 82586 performs the following housekeeping tasks:

- Updates the Actual Count field of the last Buffer Descriptor used to hold the frame just received with the number of bytes stored in its associated Data Buffer.

- Fetches the address of the next free Receive Frame Descriptor.
- Writes the address of the next free Buffer Descriptor into the next free Receive Frame Descriptor.
- Posts a 'Frame Received' interrupt status bit in the SCB.
- Interrupts the CPU.

In the event of a frame error, such as a CRC error, the 82586 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. As long as Receive Frame Descriptors and data buffers are available, the 82586 will continue to receive frames without further CPU help.

82586 NETWORK MANAGEMENT AND DIAGNOSTIC FUNCTIONS

The behavior of data communication networks is typically very complex due to their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82586 was designed in anticipation of these problems and includes a set of features for improving reliability and testability.

The 82586 reports on the following events after each frame transmitted:

- Transmission successful.
- Transmission unsuccessful; lost Carrier Sense.
- Transmission unsuccessful; lost Clear-to-Send.
- Transmission unsuccessful; DMA underrun because the system bus did not keep up with the transmission.
- Transmission unsuccessful; number of collisions exceeded the maximum allowed.

The 82586 checks each incoming frame and reports on the following errors, (if configured to 'Save Bad Frame'):

- CRC error: incorrect CRC in a well aligned frame.
- Alignment error: incorrect CRC in a misaligned frame.
- Frame too short: the frame is shorter than the configured value for minimum frame length.
- Overrun: the frame was not completely placed in memory because the system bus did not keep up with incoming data.
- Out of buffers: no memory resources to store the frame, so part of the frame was discarded.

NETWORK PLANNING AND MAINTENANCE

To perform proper planning, operation, and maintenance of a communication network, the network management entity must accumulate information on network behavior. The 82586 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Network Activity information is provided in the status of each frame transmitted. The activity indicators are:

- Number of collisions: number of collisions the 82586 experienced in attempting to transmit this frame.
- Deferred transmission: indicates if the 82586 had to defer to traffic on the link during the first transmission attempt.

Statistics registers are updated after each received frame that passes the address filtering, and is longer than the Minimum Frame Length configuration parameter.

- CRC errors: number of frames that experienced a CRC error and were properly aligned.
- Alignment errors: number of frames that experienced a CRC error and were misaligned.
- No-resources: number of correct frames lost due to lack of memory resources.
- Overrun errors: number of frame sequences lost due to DMA overrun.

The 82586 can be configured to Promiscuous Mode. In this mode it captures all frames transmitted on the Network without checking the Destination Address. This is useful in implementing a monitoring station to capture all frames for analysis.

The 82586 is capable of determining if there is a short or open circuit anywhere in the Network using the built in Time Domain Reflectometer (TDR) mechanism.

STATION DIAGNOSTICS

The chip can be configured to External Loopback. The transmitter to receiver interconnection can be placed anywhere between the 82586 and the link to locate faults, for example: the 82586 output pins, the Serial Interface Unit, the Transceiver cable, or in the Transceiver.

The 82586 has a mechanism recognizing the transceiver 'heart beat' signal for verifying the correct operation of the Transceiver's collision detection circuitry.

82586 SELF TESTING

The 82586 can be configured to Internal Loopback. It disconnects itself from the Serial Interface Unit, and any frame transmitted is received immediately. The 82586 connects the Transmit Data to the Receive Data signal and the Transmit Clock to the Receive Clock.

The Dump Command causes the chip to write over 100 bytes of its internal registers to memory.

The Diagnose command checks the exponential Backoff random number generator internal to the 82586.

CONTROLLING THE 82586

The CPU controls operation of the 82586's Command Unit (CU) and Receive Unit (RU) of the 82586 via the System Control Block.

THE COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block (CB) is associated with each Action Command.

The CU can be modeled as a logical machine that takes, at any given time, one of the following states:

- IDLE—CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- SUSPENDED—CU is not executing a command but (different from IDLE) is associated with a CB on the list.
- ACTIVE—CU is currently executing an Action Command, and points to its CB.

The CPU may affect the CU operation in two ways: issuing a CU control Command or setting bits in the COMMAND word of the Action Command.

THE RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory.

The RU is modeled as a logical machine that takes, at any given time, one of the following states:

- IDLE—RU has no memory resources and is discarding incoming frames. This is the initial RU state.
- NO-RESOURCES—RU has no memory resources and is discarding incoming frames. This state differs from the IDLE state in that RU accumulates statistics on the number of frames it had to discard.
- SUSPENDED—RU has free memory resources to store incoming frames but discard them anyway.

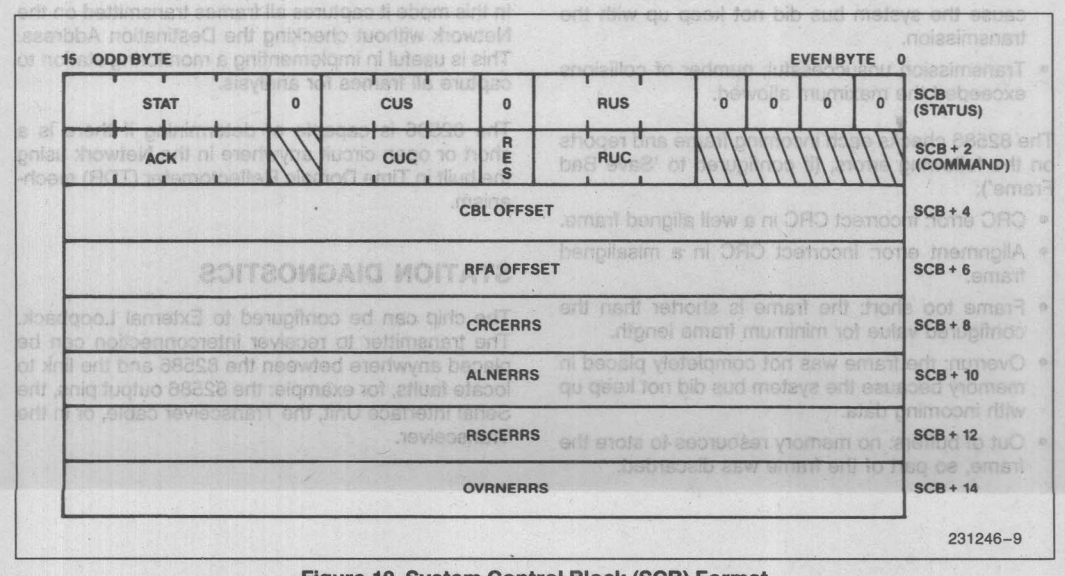


Figure 10. System Control Block (SCB) Format

- READY—RU has free memory resources and stores incoming frames.

The CPU may affect RU operation in three ways: issuing an RU Control Command, setting bits in Frame Descriptor, FD, COMMAND word of the frame currently being received, or setting EL bit of Buffer Descriptor, BD, of the buffer currently being filled.

SYSTEM CONTROL BLOCK (SCB)

The System Control Block is the communication mail-box between the 82586 and the host CPU. The SCB format is shown in Figure 10.

The host CPU issues Control Commands to the 82586 via the SCB. These commands may appear at any time during routine operation, as determined by the host CPU. After the required Control Command is setup, the CPU sends a CA signal to the 82586.

SCB is also used by the 82586 to return status information to the host CPU. After inserting the required status bits into SCB, the 82586 issues an Interrupt to the CPU.

The format is as follows:

STATUS word: Indicates the status of the 82586. This word is modified only by the 82586. Defined bits are:

CX	(Bit 15)	• A command in the CBL having its 'I' (interrupt) bit set has been executed.
FR	(Bit 14)	• A frame has been received.
CNR	(Bit 13)	• The Command Unit left the Active state.
RNR	(Bit 12)	• The Receive Unit left the Ready state.
CUS	(Bits 8–10)	• (3 bits) this field contains the status of the Command Unit. Valid values are: 0 — Idle 1 — Suspended 2 — Active 3–7 — Not Used
RUS	(Bits 4—6)	• (3 bits) this field contains the status of the Receive Unit. Valid values are: 0 — Idle 1 — Suspended 2 — No Resources 3 — Not Used 4 — Ready 5–7 — Not Used

COMMAND word: Specifies the action to be performed as a result of the CA. This word is set by the CPU and cleared by the 82586. Defined bits are:

ACK-CX	(Bit 15)	• Acknowledges the command executed event.
ACK-FR	(Bit 14)	• Acknowledges the frame received event.
ACK-CNA	(Bit 13)	• Acknowledes that the Command Unit became not ready.
ACK-RNR	(Bit 12)	• Acknowledges that the Receive Unit became not ready.
CUC	(Bits 8–10)	• (3 bits) this field contains the command to the Command Unit.
	0	• NOP (doesn't affect current state of the unit).
	1	• Start execution of the first command on the CBL. If a command is in execution, then complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET.
	2	• Resume the operation of the command unit by executing the next command. This operation assumes that the command unit has been previously suspended.
	3	• Suspend execution of commands on CBL after current command is complete.
	4	• Abort execution of commands immediately.
	5–7	• Reserved, illegal for use.
RUC	(Bits 4–6)	• (3 bits) This field contains the command to the receive unit. Valid values are:
	0	• NCP (does not alter current state of unit).
	1	• Start reception of frames. If a frame is being received, then complete reception before starting. The beginning of the RFA is contained in the RFA OFFSET.
	2	• Resume frame receiving (only when in suspended state.)
	3	• Suspend frame receiving. If a frame is being received, then complete its reception before suspending.
	4	• Abort receiver operation immediately.
	5–7	• Reserved, illegal for use.
RESET	(Bit 7)	• Reset chip (logically the same as hardware RESET).

CBL-OFFSET:

Gives the 16-bit offset address of the first command (Action Command) in the command list to be executed following CU-START. Thus, the 82586 reads this word only if the CUC field contained a CU-START Control Command.

RFA-OFFSET:

Points to the first Receive Frame Descriptor in the Receive Frame Area.

CRCERRS:

CRC Errors - contains the number of properly aligned frames received with a CRC error.

ALNERRS:

Alignment Errors - contains the number of misaligned frames received with a CRC error.

RSCERRS:

Resource Errors - records the number of correct incoming frames discarded due to lack of memory resources (buffer space or received frame descriptors).

OVRNERRS:

Overrun Errors - counts the number of received frame sequences lost because the memory bus was not available in time to transfer them.

ACTION COMMANDS

The 82586 executes a 'program' that is made up of action commands in the Command List. As shown in Figure 5, each command contains the command field, status and control fields, link to the next action command in the CL, and any command-specific parameters. This command format is called the Command Block.

The 82586 has a repertoire of 8 commands:

NOP
Setup Individual Address
Configure
Setup Multicast Address
Transmit
TDR
Diagnose
Dump

NOP

This command results in no action by the 82586, except as performed in normal command processing. It is present to aid in Command List manipulation.

NOP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• NOP = 0

LINK OFFSET: Address of next Command Block

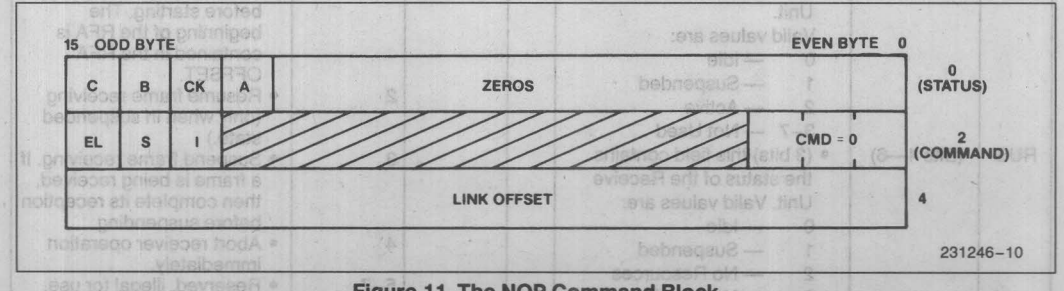


Figure 11. The NOP Command Block

IA-SETUP

This command loads the 82586 with the Individual Address. This address is used by the 82586 for recognition of Destination Address during reception and insertion of Source Address during transmission.

The IA-SETUP command includes the following fields:

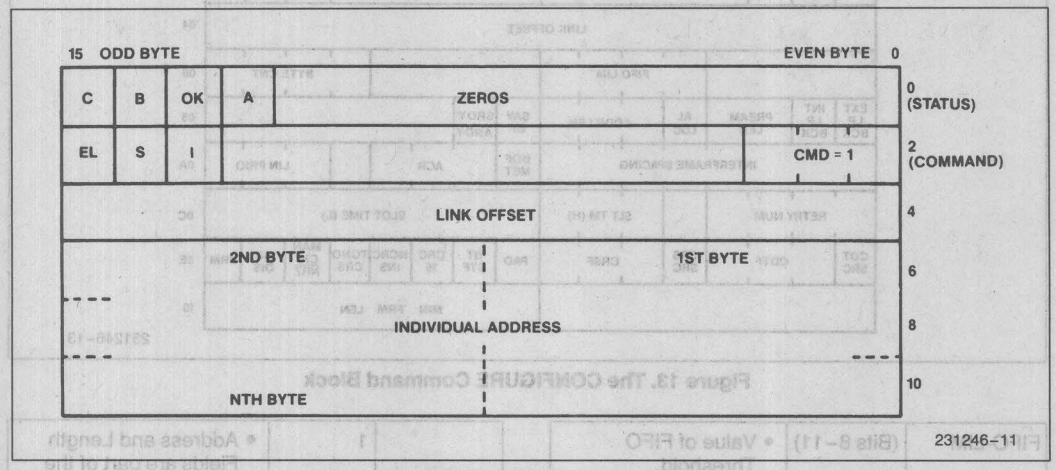


Figure 12. The IA-SETUP Command Block

STATUS word (written by 82586).

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• IA-SETUP = 1

LINK OFFSET: Address of next Command Block

INDIVIDUAL ADDRESS: Individual Address parameter

The least significant bit of the Individual Address parameter must be zero for IEEE 802.3/Ethernet. However, no enforcement of 0 is provided by the 82586. Thus, an Individual Address with least significant bit 1, is possible.

CONFIGURE

The CONFIGURE command is used to update the 82586 operating parameters.

The CONFIGURE command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• Configure = 2

LINK OFFSET: Address of next Command Block

Byte 6–7:

BYTE CNT	(Bits 0–3)	• Byte Count, Number of bytes including this one, holding the parameters to be configured. A number smaller than 4 is interpreted as 4. A number greater than 12 is interpreted as 12.

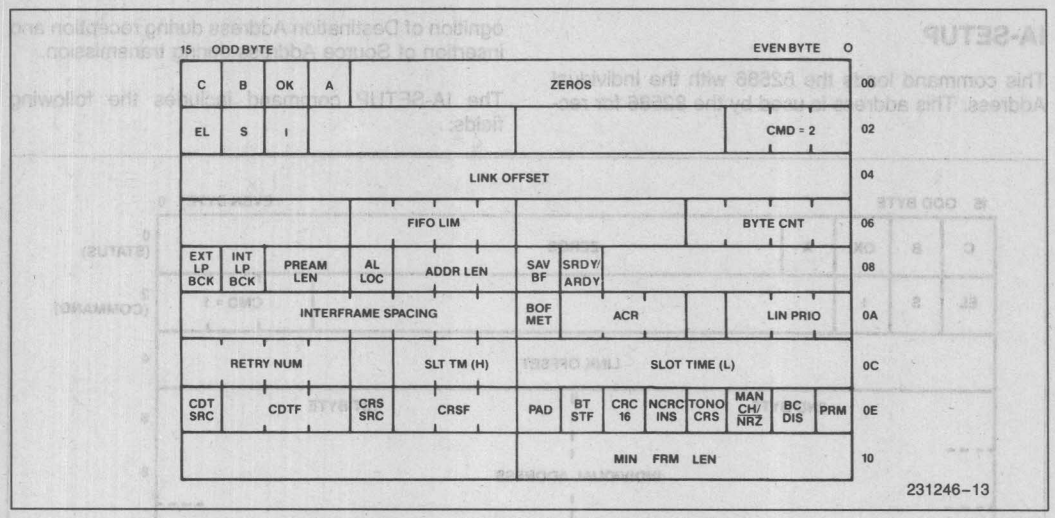


Figure 13. The CONFIGURE Command Block

FIFO-LIM	(Bits 8–11)	• Value of FIFO Threshold.

Byte 8–9:

SRDY/ARDY	(Bit 6)	
	0	• SRDY/ARDY pin operates as ARDY (internal synchronization).
	1	• SRDY/ARDY pin operates as SRDY (external synchronization).
SAV-BF	(Bit 7)	
	0	• Received bad frames are not saved in memory.
	1	• Received bad frames are saved in memory.
ADD-LEN	(Bits 8–10)	• Number of address byes. NOTE: 7 is interpreted as 0.
AL-LOC	(Bit 11)	
	0	• Address and Length Fields separated from data and associated with Transmit Command Block or Receive Frame Descriptor. For transmitted Frame, Source Address is inserted by the 82586.
	1	• Address and Length Fields are part of the Transmit/Receive data buffers, including Source Address (which is not inserted by the 82586).
PREAM-LEN	(Bits 12–13)	• Preamble Length including Beginning of Frame indicator: 00 - 2 bytes 01 - 4 bytes 10 - 8 bytes 11 - 16 bytes
INT-LPBCK	(Bit 14)	• Internal Loopback
EXT-LPBCK	(Bit 15)	• External Loopback. NOTE: Bits 14 and 15 configured to 1, cause Internal Loopback.

Byte 10–11:

LIN-PRIO	(Bits 0–2)	• Linear Priority
ACR	(Bits 4–6)	• Accelerated Contention Resolution (Exponential Priority)
BOF-MET	(Bit 7)	• Exponential Backoff Method 0 - IEEE 802.3/Ethernet 1 - Alternate Method

INTER FRAME SPACING	(Bits 8–15)	• Number indicating the Interframe Spacing in TxC period units.

Byte 12–13:

SLOT-TIME (L)	(Bits 0–7)	• Slot Time Number, Low Byte
SLT-TM (H)	(Bits 8–10)	• Slot Time Number, High Bits
RETRY-NUM	(Bits 12–15)	• Maximum Number of Transmission Retries on Collisions

Byte 14–15:

PRM	(Bit 0)	• Promiscuous Mode
BC-DIS	(Bit 1)	• Broadcast Disable
MANCH/$\overline{\text{NRZ}}$	(Bit 2)	• Manchester or NRZ • Encoding/Decoding
	0	• NRZ
	1	• Manchester
TONO-CRS	(Bit 3)	• Transmit on No Carrier Sense
	0	• Cease Transmission if CRS Goes Inactive During Frame Transmission
	1	• Continue Transmission Even if no Carrier Sense
NCRC-INS	(Bit 4)	• No CRC Insertion
CRC-16	(Bit 5)	• CRC Type:
	0	• 32 bit Autodin II CRC Polynomial
	1	• 16 bit CCITT CRC Polynomial
BT-STF	(Bit 6)	• Bitstuffing:
	0	• End of Carrier Mode (Ethernet)
	1	• HDLC like Bitstuffing Mode
PAD	(Bit 7)	• Padding
	0	• No Padding
	1	• Perform Padding by Transmitting Flags for Remainder of Slot Time
CRSF	(Bits 8–9)	• Carrier Sense Filter in Bit Times
CRS-SRC	(Bit 11)	• Carrier Sense Source
	0	• External
	1	• Internal

CDTF	(Bits 12–14	• Collision Detect Filter in Bit Times
CDT-SRC	(Bit 15)	• Collision Detect Source
	0	• External
	1	• Internal

Byte 16:

MIN-FRM-	(Bits 0–7)	• Minimum Number of Bytes in a Frame

CONFIGURATION DEFAULTS

The default values of the configuration parameters are compatible with the IEEE 802.3/Ethernet Standards. RESET configures the 82586 according to the defaults shown in Table 2.

Table 2. 82586 Default Values

Preamble Length (Bytes)	=	8
Address Length (Bytes)	=	6
Broadcast Disable	=	0
CRC-16/CRC-32	=	0
No CRC Insertion	=	0
Bitstuffing/EOC	=	0
Padding	=	0
Min-Frame-Length (Bytes)	=	64
Interframe Spacing (Bits)	=	96
Slot Time (Bits)	=	512
Number of Retries	=	15
Linear Priority	=	0
Accelerated Contention Resolution	=	0
Exponential Backoff Method	=	0
Manchester/$\overline{\text{NRZ}}$	=	0
Internal CRS	=	0
CRS Filter	=	0
Internal CDT	=	0
CDT Filter	=	0
Transmit On No CRS	=	0
FIFO THRESHOLD	=	8
SRDY/$\overline{\text{ARDY}}$	=	0
Save Bad Frame	=	0
Address/Length Location	=	0
INT Loopback	=	0
EXT Loopback	=	0
Promiscuous Mode	=	0

1

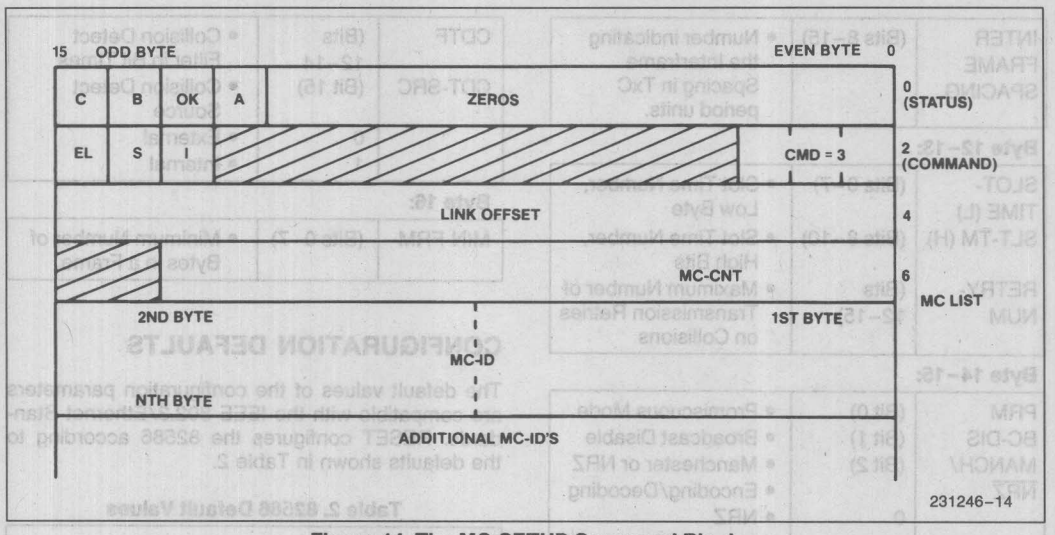


Figure 14. The MC-SETUP Command Block

MC-SETUP

This command sets up the 82586 with a set of Multicast Addresses. Subsequently, incoming frames with Destination Addresses from this set are accepted.

The MC-SETUP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• MC-SETUP = 3

LINK OFFSET: Address of next Command Block

MC-CNT: A 14-bit field indicating the number of bytes in the MC-LIST field. MC-CNT is truncated to the nearest multiple of Address Length (in bytes). Issuing a MC-SETUP command with MC-CNT = 0 disables reception of any incoming frame with a Multicast Address.

MC-LIST: A list of Multicast Addresses to be accepted by the 82586. Note that the most significant byte of an address is followed immediately by the least significant byte of the next address. Note also that the least significant bit of each Multicast Address in the set must be a one.

The Transmit-Byte-Machine maintains a 64-bit HASH table used for checking Multicast Addresses during reception.

An incoming frame is accepted if it has a Destination Address whose least significant bit is a one, and after hashing points to a bit in the HASH table whose value is one. The hash function is selecting bits 2 to 7 of the CRC register. RESET causes the HASH table to become all zeros.

After the Transmit-Byte-Machine reads a MC-SETUP command from TX-FIFO, it clears the HASH table and reads the bytes in groups whose length is determined by the ADDRESS length. Each group is hashed using CRC logic and the bit in the HASH table to which bits 2–7 of the CRC register point is set to one. A group that is not complete has no effect on the HASH table. Transmit-Byte-Machine notifies CU after completion.

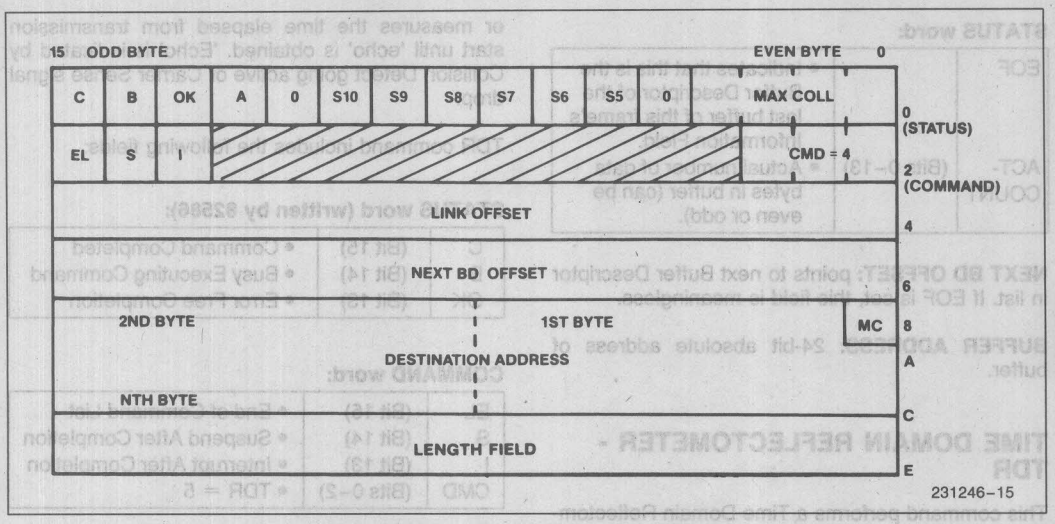


Figure 15. The Transmit Command Block

TRANSMIT

The TRANSMIT command causes transmission (and if necessary retransmission) of a frame.

TRANSMIT CB includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
A	(Bit 12)	• Command Aborted
S10	(Bit 10)	• No Carrier Sense signal during transmission (between beginning of Destination Address and end of Frame Check Sequence).
S9	(Bit 9)	• Transmission unsuccessful (stopped) due to loss of Clear-to-Send signal.
S8	(Bit 8)	• Transmission unsuccessful (stopped) due to DMA underrun, (i.e. data not supplied from the sytem for transmission).
S7	(Bit 7)	• Transmission had to Defer to traffic on the link.
S6	(Bit 6)	• Heart Beat, indicates that during Interframe Spacing period after the previous transmission, a pulse was detected on the Collision Detect pin.
S5	(Bit 5)	• Transmission attempt stopped due to number of collisions exceeding the maximum number of retries.
MAX-COLL	(Bits 3–0)	• Number of Collisions experienced by this frame. S5 = 1 and MAX-COLL = 0 indicates that there were 16 collisions.

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• TRANSMIT = 4

LINK OFFSET: Address of next Command Block

TBD OFFSET: Address of list of buffers holding the information field. TBD-OFFSET = 0FFFFH indicates that there is no Information field.

DESTINATION ADDRESS: Destination Address of the frame.

LENGTH FIELD: Length field of the frame.

STATUS word:

EOF		• Indicates that this is the Buffer Descriptor of the last buffer of this frame's Information Field.
ACT-COUNT	(Bits 0–13)	• Actual number of data bytes in buffer (can be even or odd).

NEXT BD OFFSET: points to next Buffer Descriptor in list. If EOF is set, this field is meaningless.

BUFFER ADDRESS: 24-bit absolute address of buffer.

TIME DOMAIN REFLECTOMETER - TDR

This command performs a Time Domain Reflectometer test on the serial link. By performing the command, the user is able to identify shorts or opens and their location. Along with transmission of 'All Ones,' the 82586 triggers an internal timer. The timer measures the time elapsed from transmission start until 'echo' is obtained. 'Echo' is indicated by Collision Detect going active or Carrier Sense signal drop.

TDR command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• TDR = 5

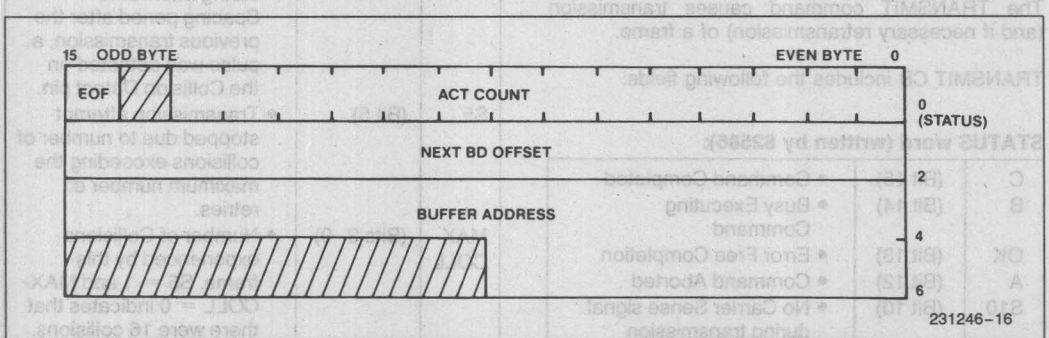


Figure 16. The Transmit Buffer Description

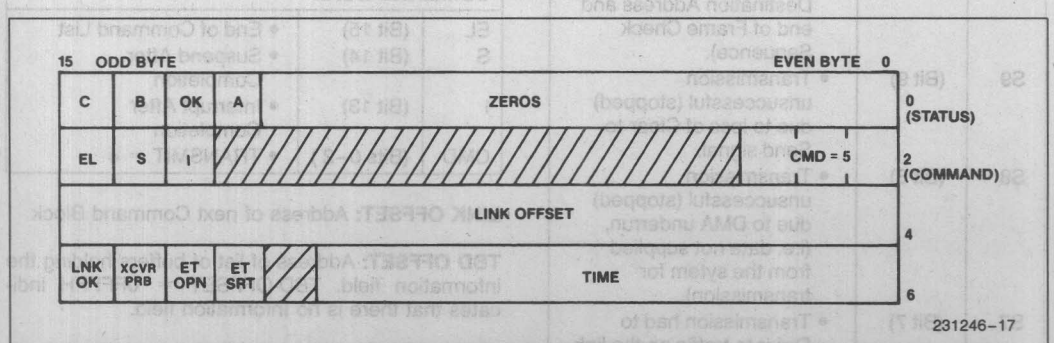


Figure 17. The TDR Command Block

LINK OFFSET: Address of next Command Block

RESULT word:

LNK-OK	(Bit 15)	• No Link Problem Identified
XCVR-PRB	(Bit 14)	• Transceiver Cable Problem identified (valid only in the case of a Transceiver that does not return Carrier Sense during transmission).
ET-OPN	(Bit 13)	• Open on the link identified (valid only in the case of a Transceiver that returns Carrier Sense during transmission).
ET-SRT	(Bit 12)	• Short on the link identified (valid only in the case of a Transceiver that returns Carrier Sense during transmission).
TIME	(Bits 0–10)	• Specifying the distance to a problem on the link (if one exists) in transmit clock cycles.

DUMP

This command causes the contents of over a hundred bytes of internal registers to be placed in memory. It is supplied as a self diagnostic tool, as well as to supply registers of interest to the user.

DUMP command includes the following fields:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• DUMP = 6

LINK OFFSET: Address of next Command Block

BUFFER OFFSET: This word specifies the offset portion of the memory address which points to the top of the buffer allocated for the dumped registers contents. The length of the buffer is 170 bytes.

DUMP AREA FORMAT

Figure 18 shows the format of the DUMP area. The fields are as follows:

Bytes 00H to 0AH: These bytes correspond to the 82586 CONFIGURE command field.

Bytes 0CH to 11H: The Individual Address Register content. IARO is the Individual Address least significant byte.

Bytes 12H to 13H: Status word of last command block (only bits 0–13).

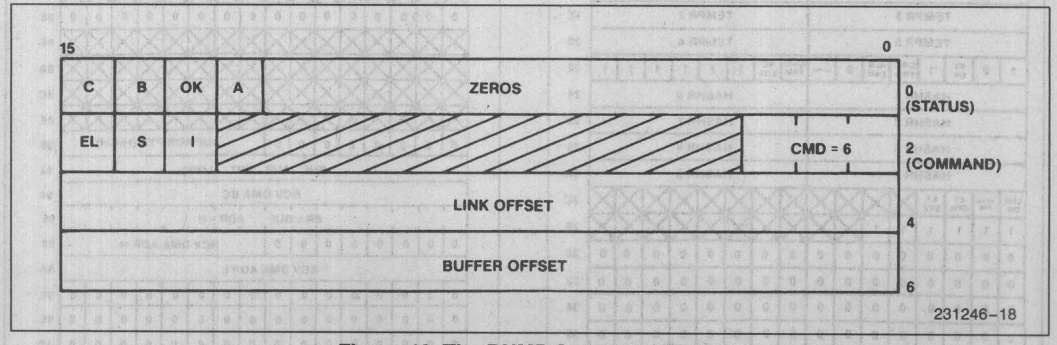


Figure 18. The DUMP Command Block

Bytes 14H to 17H: Content of the Transmit CRC generator. TXCRCRO is the least significant byte. The contents are dependent on the activity before the DUMP command:

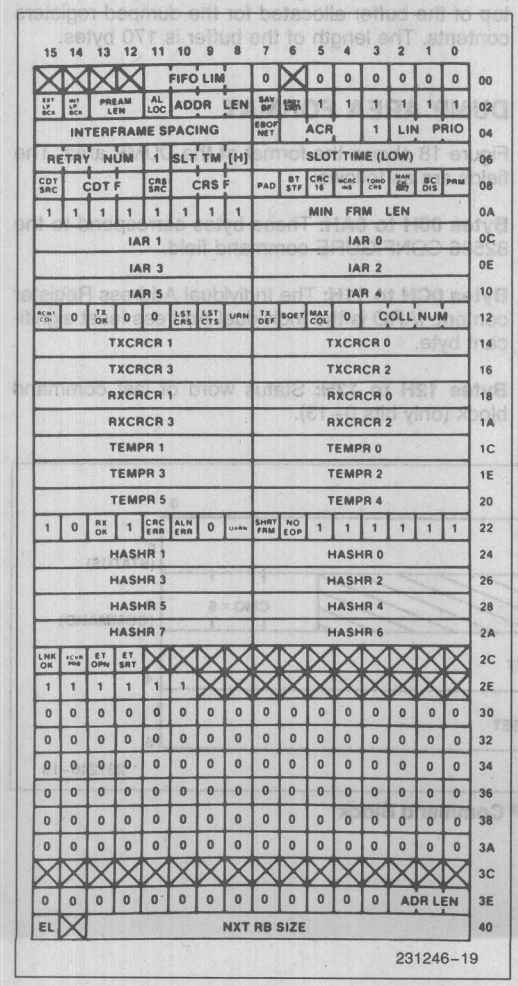
After RESET - 'All Ones.'

After successful transmission - 'All Zeros'.

After MC-SETUP command - Generated CRC value of the last MC address, on MC-LIST.

After unsuccessful transmission, depends on where it stopped.

NOTE:
For 16-bit CRC only TXCRCRO and TXCRCR1 are valid.



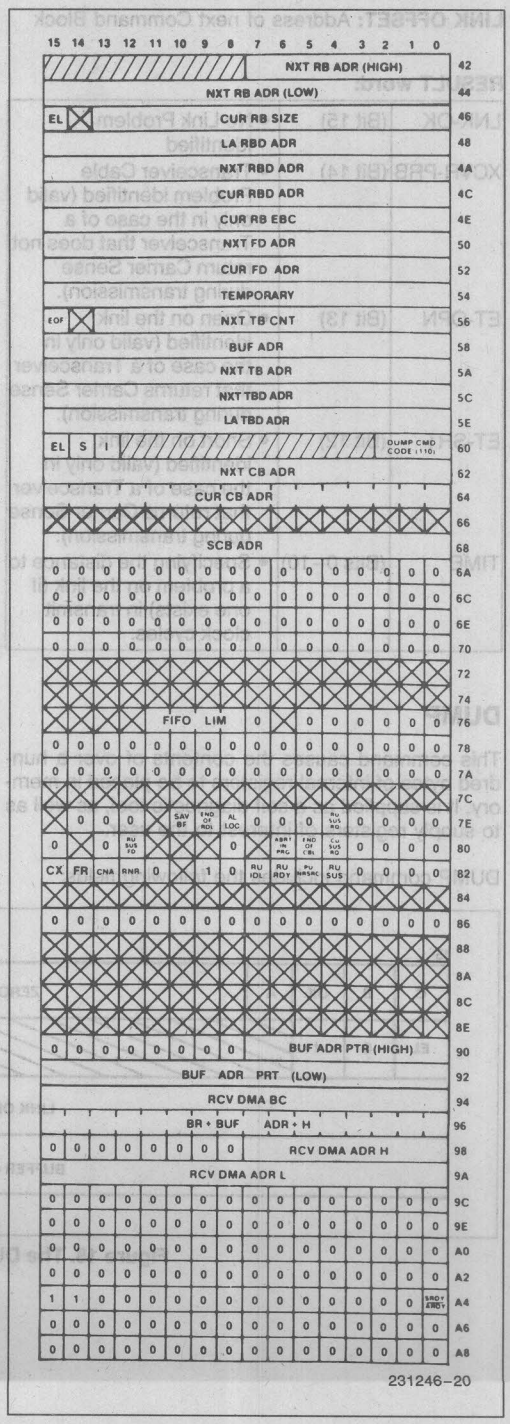


Figure 19. The DUMP Area

Bytes 18H to 1BH: Contents of Receive CRC Checker. RXCRCRO is the least significant byte. The contents are dependent on the activity performed before the DUMP command:

After RESET - 'All Ones.'

After good frame reception—

1. For CRC-CCITT - 0ID0FH
2. For CRC-Autodin-II - C704DD7BH

After Bad Frame reception - corresponds to the received information.

After reception attempt, i.e. unsuccessful check for address match, corresponds to the CRC performed on the frame address.

NOTE:
Any frame on the serial link modifies this register contents.

Bytes 1CH to 21H: Temporary Registers.

Bytes 22H to 23H: Receive Status Register. Bits 6, 7, 8, 10, 11 and 13 assume the same meaning as corresponding bits in the Receive Frame Descriptor Status field.

Bytes 24H to 2BH: HASH TABLE.

Bytes 2CH to 2DH: Status bits of the last time TDR command that was performed.

NXT-RB-SIZE: Let N be the last buffer of the last received frame, then NXT-RB-SIZE is the number of bytes of available in the N + 1 buffer. EL - The EL bit of the Receive Buffer Descriptor.

NXT-RB-ADR: Let N be the last Receive Buffer used, then NXT-RB-ADR is the BUFFER-ADDRESS field in the N + 1 Receive-Buffer Descriptor, i.e. the pointer to the N + 1 Receive Buffer.

CUR-RB-SIZE: The number of bytes in the last buffer of the last received frame. EL - The EL bit of the last buffer in the last received frame.

LA-RBD-ADR: Look Ahead Buffer Descriptor, i.e. the pointer to N + 2 Receiver Buffer Descriptor.

NXT-RBD-ADR: Next Receive Buffer Descriptor Address. Similar to LA-RBD-ADR but points to N + 1 Receive Buffer Descriptor.

CUR-RBD-ADR: Current Receive Buffer Descriptor Address. Similar to LA-RBD-ADR, but point to Nth Receive Buffer Descriptor.

CUR-RB-EBC: Current Receive Buffer Empty Byte Count Let N be the currently used Receive Buffer. Then CUR-RB-EBC indicates the Empty part of the buffer, i.e. the ACT-COUNT of buffer N is given by the difference between its SIZE and the CUR-RB-EBC.

NXT-FD-ADR: Next Frame Descriptor Address. Define N as the last Receive Frame Descriptor with bits C = 1 and B = 0, then NXT-FD-ADR is the address of N + 2 Receive Frame Descriptor (with B = C = 0) and is equal to the LINK-ADDRESS field in N + 1 Receive Frame Descriptor.

CUR-FD-ADR: Current Frame Descriptor Address. Similar to next NXT-FD-ADR but refers to N + 1 Receive Frame Descriptor (with B = 1, C = 0).

Bytes 54H to 55H: Temporary register.

NXT-TB-CNT: Next Transmit Buffer Count. Let N be the last transmitted buffer of the TRANSMIT command executed recently, the NXT-TB-CNT is the ACT-COUNT field in the Nth Transmit Buffer Descriptor. EOF - Corresponds to the EOF bit of the Nth Transmit Buffer Descriptor. EOF = 1 indicates that the last buffer accessed by the 82586 during Transmit was the last Transmit Buffer in the data buffer chain associated with the Transmit Command.

BUF-ADR: Buffer Address. The BUF-PTR field in the DUMP-STATUS Command Block.

NXT-TB-AD-L: Next Transmit Buffer Address Low. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then NXT-TB-AD-L are the two least significant bytes of the Nth buffer address.

LA-TB-ADR: Look Ahead Transmit Buffer Descriptor Address. Let N be the last Transmit Buffer in the transmit buffer chain of the TRANSMIT Command performed recently, then LA-TBD-ADR is the NEXT-BD-ADDRESS field of the Nth Buffer Descriptor.

NXT-TBD-ADR: Next Transmit Buffer Descriptor Address. Similar in function to LA-TBD-ADR but related to Transmit Buffer Descriptor N-1. Actually, it is the address of Transmit Buffer Descriptor N.

Bytes 60H, 61H: This is a copy of the 2nd word in the DUMP-STATUS command presently executing.

NXT-CB-ADR: Next Command Block Address. The LINK-ADDRESS field in the DUMP Command Block presently executing. Points to the next command.

CUR-CB-ADR: Current Command Block Address. The address of the DUMP Command Block currently executing.

SCB-ADR: Offset of the System Control Block (SCB).

Bytes 7EH, 7FH:

RU-SUS-RQ (Bit 4) - Receive Unit Suspend Request.

Bytes 80H, 81H:

CU-SUS-RQ (Bit 4) - Command Unit Suspend Request.

END-OF-CBL (Bit 5) - End of Command Block List. If "1" indicates that DUMP-STATUS is the last command in the command chain.

ABRT-IN-PROG (Bit 6) - Command Unit Abort Request.

RU-SUS-FD (Bit 12) - Receive Unit Suspend Frame Descriptor Bit. Assume N is the Receive Frame Descriptor used recently, then RU-SUS-FD is equivalent to the S bit of N + 1 Receive Frame Descriptor.

Bytes 82H, 83H:

RU-SUS (Bit 4) - Receive Unit in SUSPENDED state.

RU-NRSRC (Bit 5) - Receive Unit in NO RESOURCES state.

RU-RDY (Bit 6) - Receive Unit in READY state.

RU-IDL (Bit 7) - Receive Unit in IDLE state.

RNR (Bit 12) - RNR Interrupt in Service bit.

CNA (Bit 13) - CNA Interrupt in Service bit.

FR (Bit 14) - FR Interrupt in Service bit.

CX (Bit 15) - CX Interrupt in Service bit.

Bytes 90H to 93H:

BUF-ADR-PTR - Buffer pointer is the absolute address of the bytes following the DUMP Command block.

Bytes 94H to 95H:

RCV-DMA-BC - Receive DMA Byte Count. This field contains number of bytes to be transferred during the next Receive DMA operation. The value depends on AL-LOCation configuration bit.

1. If AL-LOCation = 0 then RCV-DMA-BC = (2 times ADDR-LEN plus 2) if the next Receive Frame Descriptor has already been fetched.
2. If AL-LOCation = 1 then it contains the size of the next Receive Buffer.

BR + BUF − PTR + 96H - Sum of Base Address plus BUF − PTR field and 96H.

RCV-DMA-ADR - Receive DMA absolute Address. This is the next RCV-DMA start address. The value depends on AL-LOCation configuration bit.

1. If AL-LOCation = 0, then RCV-DMA-ADR is the Destination Address field located in the next Receive Frame Descriptor.
2. If AL-LOCation = 1, then RCV-DMA-ADR is the next Receive Data Buffer Address.

The following nomenclature has been used in the DUMP table:

0	• The 82586 writes zero in this location.
1	• The 82586 writes one in this location.
X	• The 82586 writes zero or one in this location.
///	• The 82586 copies this location from the corresponding position in the memory structure.

DIAGNOSE

The DIAGNOSE Command triggers an internal self test procedure of backoff related registers and counters.

The DIAGNOSE command includes the following:

STATUS word (written by 82586):

C	(Bit 15)	• Command Completed
B	(Bit 14)	• Busy Executing Command
OK	(Bit 13)	• Error Free Completion
FAIL	(Bit 11)	• Indicates that the Self Test Procedured Failed

COMMAND word:

EL	(Bit 15)	• End of Command List
S	(Bit 14)	• Suspend After Completion
I	(Bit 13)	• Interrupt After Completion
CMD	(Bits 0–2)	• DIAGNOSE = 7

LINK OFFSET: Address of next Command Block.

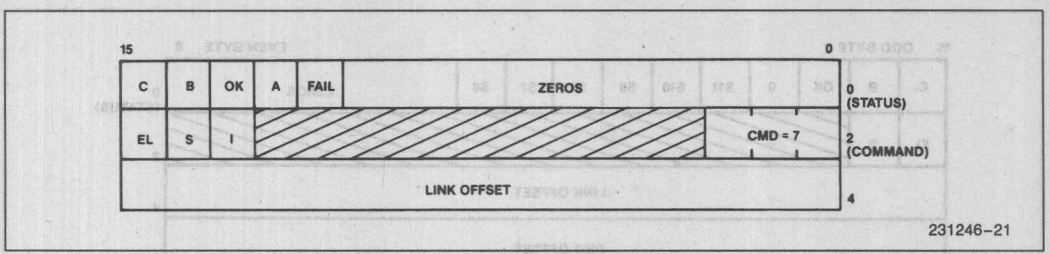


Figure 20. The DIAGNOSE Command Block

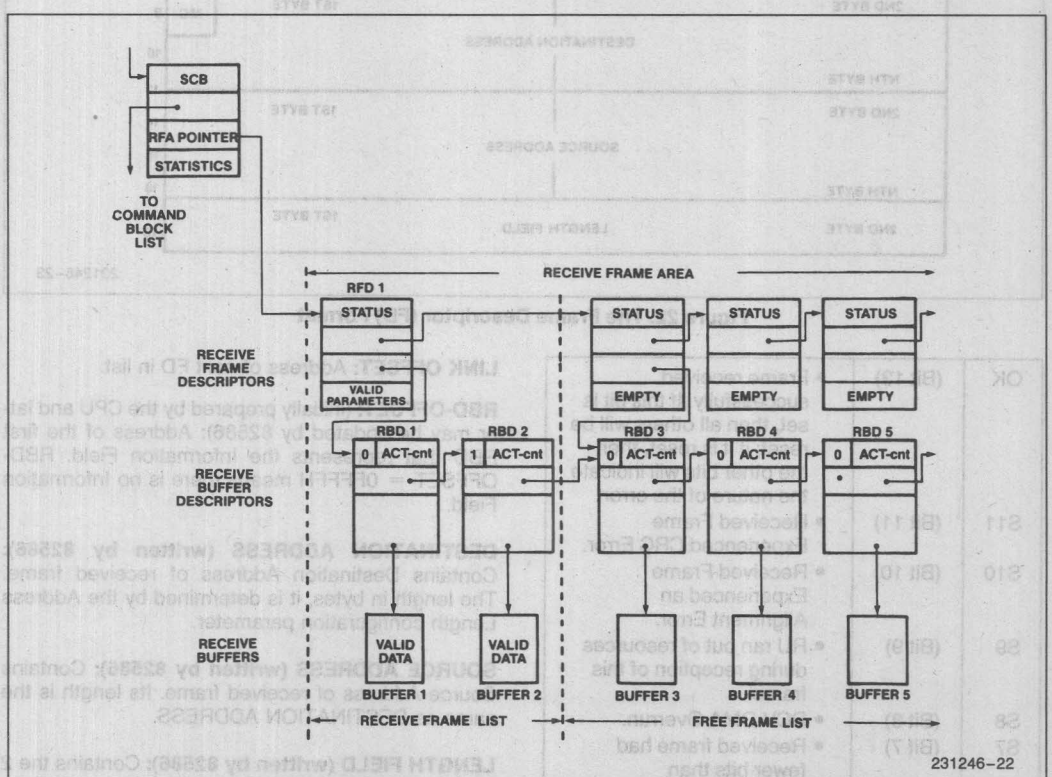


Figure 21. The Receive Frame Area

RECEIVE FRAME AREA (RFA)

The Receive Frame Area, RFA, is prepared by the host CPU, data is placed into the RFA by the 82586 as frames are received. RFA consists of a list of Receive Frame Descriptors (FD), each of which is associated with a frame. RFA-OFFSET field of SCB points to the first FD of the chain; the last FD is identified by the End-of-Listing flag (EL). See Figure 21.

FRAME DESCRIPTOR (FD) FORMAT

The FD includes the following fields:

STATUS word (set by the 82586):

C	(Bit 15)	• Completed Storing Frame.
B	(Bit 14)	• FD was Consumed by RU.

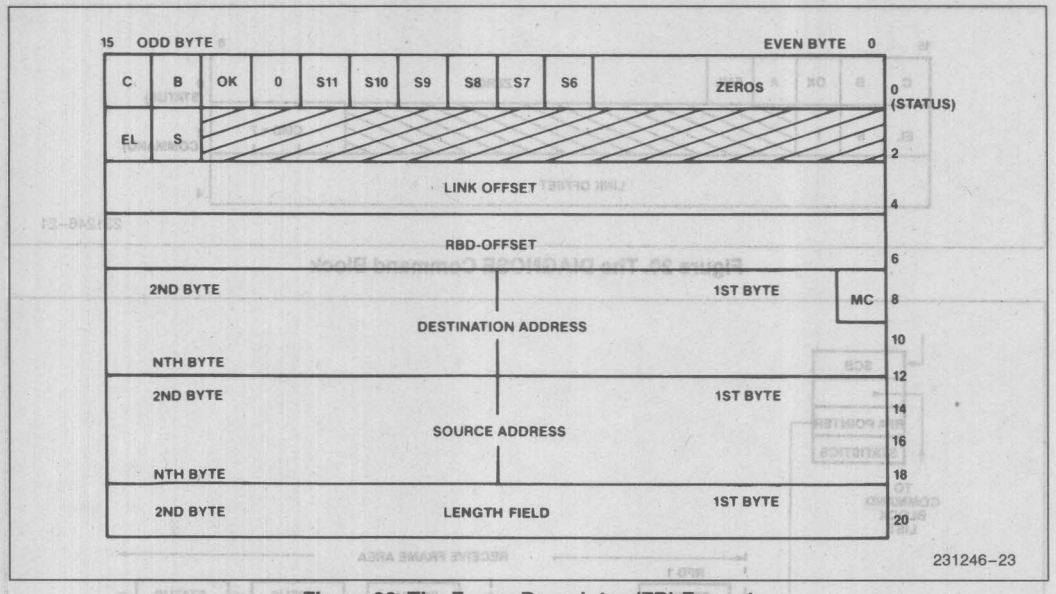


Figure 22. The Frame Descriptor (FD) Format

OK	(Bit 13)	• Frame received successfully. If this bit is set, then all others will be reset; if it is reset, then the other bits will indicate the nature of the error.
S11	(Bit 11)	• Received Frame Experienced CRC Error.
S10	(Bit 10)	• Received Frame Experienced an Alignment Error.
S9	(Bit 9)	• RU ran out of resources during reception of this frame.
S8	(Bit 8)	• RCV-DMA Overrun.
S7	(Bit 7)	• Received frame had fewer bits than configured Minimum Frame Length.
S6	(Bit 6)	• No EOF flag detected (only when configured to Bitstuffing).

COMMAND word:

EL	(Bit 15)	• Last FD in the List.
S	(Bit 14)	• RU should be suspended after receiving this frame.

LINK OFFSET: Address of next FD in list.

RBD-OFFSET: (initially prepared by the CPU and later may be updated by 82586): Address of the first RBD that represents the Information Field. RBD-OFFSET = 0FFFFH means there is no Information Field.

DESTINATION ADDRESS (written by 82586): Contains Destination Address of received frame. The length in bytes, it is determined by the Address Length configuration parameter.

SOURCE ADDRESS (written by 82586): Contains Source Address of received frame. Its length is the same as DESTINATION ADDRESS.

LENGTH FIELD (written by 82586): Contains the 2 byte Length or Type Field of received frame.

RECEIVE BUFFER DESCRIPTOR FORMAT

The Receive Buffer Descriptor (RBD) holds information about a buffer; size and location, and the means for forming a chain of RBDs, (forward pointer and end-of-frame indication).

The Buffer Descriptor contains the following fields.

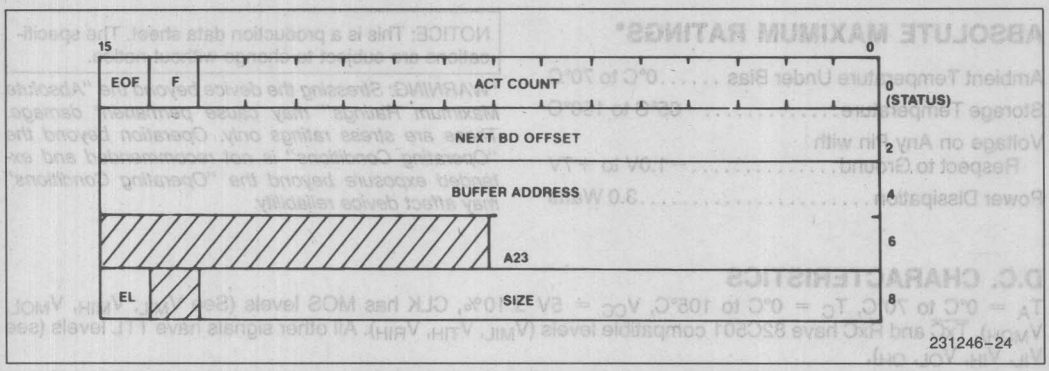


Figure 23. The Receive Buffer Descriptor (RBD) Format

STATUS word (written by the 82586).

EOF	(Bit 15)	• Last buffer in received frame.
F	(Bit 14)	• ACT COUNT field is valid.
ACT COUNT	(Bits 0–13)	• Number of bytes in the buffer that are actually occupied.

NEXT RBD OFFSET: Address of next BD in list of BD's.

BUFFER ADDRESS: 24-bit absolute address of buffer.

EL/SIZE:

EL	(BIT 15)	• Last BD in list.
SIZE	(Bits 0–13)	• Number of bytes the buffer is capable of holding.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature.....−65°C to 150°C
Voltage on Any Pin with
Respect to Ground.....−1.0V to +7V
Power Dissipation.....3.0 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

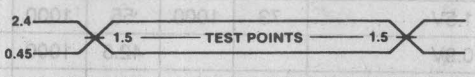
T_A = 0°C to 70°C, T_C = 0°C to 105°C, V_{CC} = 5V ±10%, CLK has MOS levels (See V_{MIL}, V_{MIH}, V_{MOL}, V_{MOH}). $\overline{TxC}$ and $\overline{RxC}$ have 82C501 compatible levels (V_{MIL}, V_{TIH}, V_{RIH}). All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, $_{OH}$).

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (TTL)	−0.5	+0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 2.5 mA
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} − 400 μA
V_{MIL}	Input Low Voltage (MOS)	−0.5	0.6	V	
V_{MIH}	Input High Voltage (MOS)	3.9	V_{CC} + 0.5	V	
V_{TIH}	Input High Voltage ($\overline{TxC}$)	3.3	V_{CC} + 0.5	V	
V_{RIH}	Input High Voltage ($\overline{RxC}$)	3.0	V_{CC} + 0.5	V	
V_{MOL}	Output Low Voltage (MOS)		0.45	V	I_{OL} 2.5 mA
V_{MOH}	Output High Voltage (MOS)	V_{CC} − 0.5		V	I_{OH} − 400 μA
I_{LI}	Input Leakage Current		±10	μA	$0 \le V_{IN} \le V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0.45 \le V_{OUT} \le V_{CC}$
C_{IN}	Capacitance of Input Buffer		10	pF	FC = 1 MHz
C_{OUT}	Capacitance of Output Buffer		20	pF	FC = 1 MHz
I_{CC}	Power Supply Current		550 450	mA	T_A = 0°C T_A = 70°C

SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

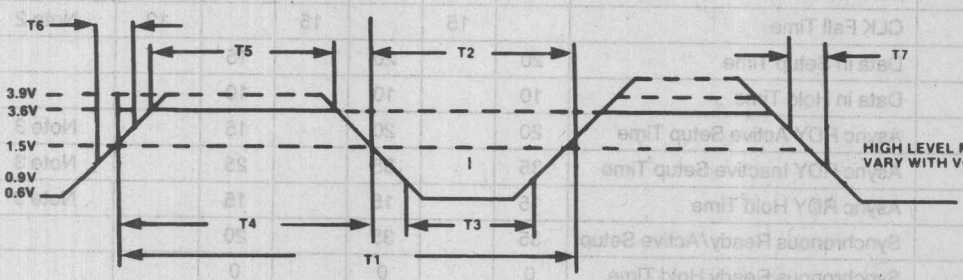
T_A = 0°C to 70°C, T_C = 0°C to 105°C, V_{CC} = 5V ±10%. Figures 24 and 25 define how the measurements should be done.

INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS



AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0

Figure 24. TTL Input/Output Voltage Levels for Timing Measurements



MOS I/O measurements are taken at 0.1 and 0.9 of the voltage swing

Figure 25. System Clock CMOS Input Voltage Levels for Timing Measurements

INPUT TIMING REQUIREMENTS*

Symbol	Parameter	82586-6 (6 MHz)		82586 (8 MHz)		82586-10 (10 MHz)		Comments
		Min	Max	Min	Max	Min	Max	
T1	CLK Cycle Period	166	2000	125	2000	100	200	
T2	CLK Low Time at 1.5V	73	1000	55	1000	44	1000	
T3	CLK Low Time at 0.9V			42.5	1000	42.5	1000	
T4	CLK High Time at 1.5V	73		55		44		
T5	CLK High Time at 3.6V			42.5		42.5		
T6	CLK Rise Time		15		15		12	Note 1
T7	CLK Fall Time		15		15		12	Note 2
T8	Data in Setup Time	20		20		15		
T9	Data in Hold Time	10		10		10		
T10	Async RDY Active Setup Time	20		20		15		Note 3
T11	Async RDY Inactive Setup Time	35		35		25		Note 3
T12	Async RDY Hold Time	15		15		15		Note 3
T13	Synchronous Ready/Active Setup	35		35		20		
T14	Synchronous Ready Hold Time	0		0		0		
T15	HLDA Setup Time	20		20		20		Note 3
T16	HLDA Hold Time	10		10		5		Note 3
T17	Reset Setup Time	20		20		20		Note 3
T18	Reset Hold Time	10		10		10		Note 3
T19	CA Pulse Width	1 T1		1 T1		1 T1		
T20	CA Setup Time	20		20		20		Note 3
T21	CA Hold Time	10		10		10		Note 3

OUTPUT TIMINGS**

Symbol	Parameter	Min	Max	Min	Max	Min	Max	Comments
T22	DT/R Valid Delay	0	60	0	60	0	44	
T23	$\overline{WR}$, $\overline{DEN}$ Active Delay	0	70	0	70	0	56	
T24	$\overline{WR}$, $\overline{DEN}$ Inactive Delay	10	65	10	65	10	45	
T25	Int. Active Delay	0	85	0	85	0	70	Note 4
T26	Int. Inactive Delay	0	85	0	85	0	70	Note 4
T27	Hold Active Delay	0	85	0	85	0	70	Note 4
T28	Hold Inactive Delay	0	85	0	85	0	70	Note 4
T29	Address Valid Delay	0	55	0	55	0	50	
T30	Address Float Delay	0	50	0	50	12	50	
T31	Data Valid Delay	0	55	0	55	0	50	Note 7
T32	Data Hold Time	0		0		0		
T33	Status Active Delay	10	60	10	60	10	45	

OUTPUT TIMINGS** (Continued)

Symbol	Parameter	82582-6 (6 MHz)		82586 (8 MHz)		82586-10 (10 MHz)		Comments
		Min	Max	Min	Max	Min	Max	
T34	Status Inactive Delay	10	70	10	70	10	50	Note 8
T35	ALE Active Delay	0	45	0	45	0	35	Note 5
T36	ALE Inactive Delay	0	45	0	45	0	37	Note 5
T37	ALE Width	T2–10		T2–10		T2–10		Note 5
T38	Address Valid to ALE Low	T2–40		T2–30		T2–25		
T39	Address Hold to ALE Inactive	T4–10		T4–10		T4–10		
T40	$\overline{RD}$ Active Delay	10	95	10	95	10	95	
T41	$\overline{RD}$ Inactive Delay	10	70	10	70	10	70	
T42	$\overline{RD}$ Width	2T1–50		2T1–50		2T1–46		
T43	Address Float to $\overline{RD}$ Active	10		10		0		
T44	$\overline{RD}$ Inactive to Address Active	T1–40		T1–40		T1–34		
T45	$\overline{WR}$ Width	2T1–40		2T1–40		2T1–34		
T46	Data Hold After $\overline{WR}$	T2–25		T2–25		T2–25		
T47	Control Inactive After Reset	0	60	0	60	0	60	Note 6

*All units are in ns.
**CL on all outputs is 20–200 pF unless otherwise specified.

NOTES:
1. 1.0V to 3.5V
2. 3.5V to 1.0V
3. To guarantee recognition at next clock
4. CL = 50 pF
5. CL = 100 pF
6. Affects:
   MIN MODE: $\overline{RD}$, $\overline{WR}$, DT/$\overline{R}$, $\overline{DEN}$
   MAX MODE: $\overline{S0}$, $\overline{S1}$
7. High address lines (A16–A24, BHE) become valid one clock before T1 only on first memory cycle after the 82586 acquired the bus.
8. $\overline{S1}$, S0 go inactive just prior to T4.

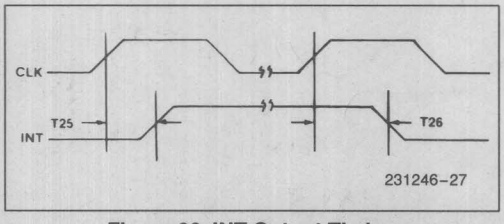


Figure 26. INT Output Timing

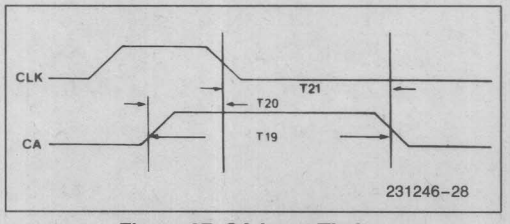


Figure 27. CA Input Timing

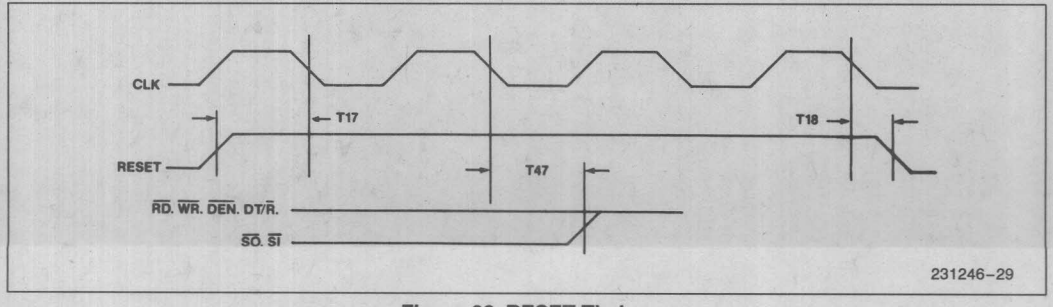


Figure 28. RESET Timing

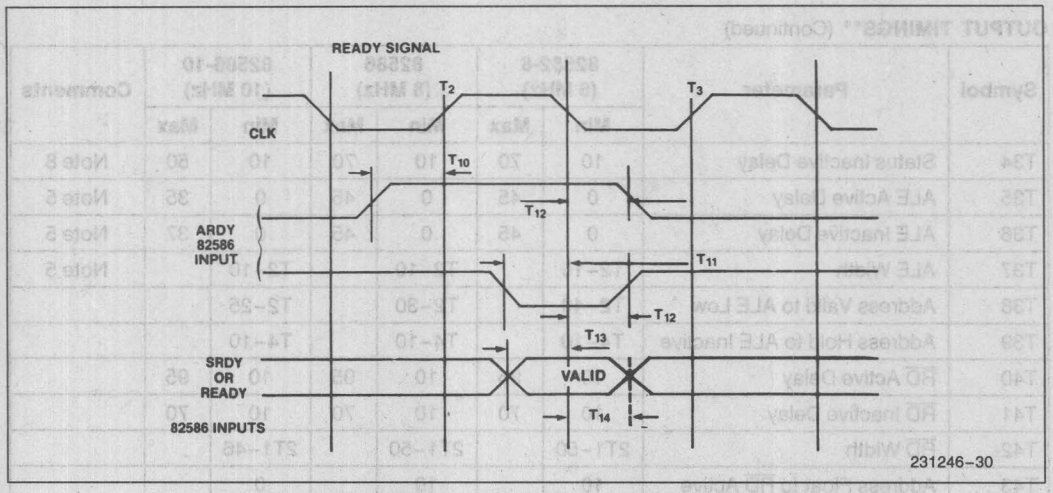


Figure 29. ARDY and SRDY Timings Relative to CLK

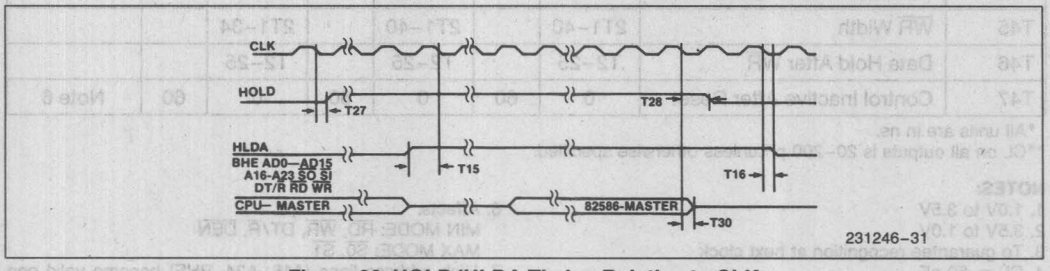


Figure 30. HOLD/HLDA Timing Relative to CLK

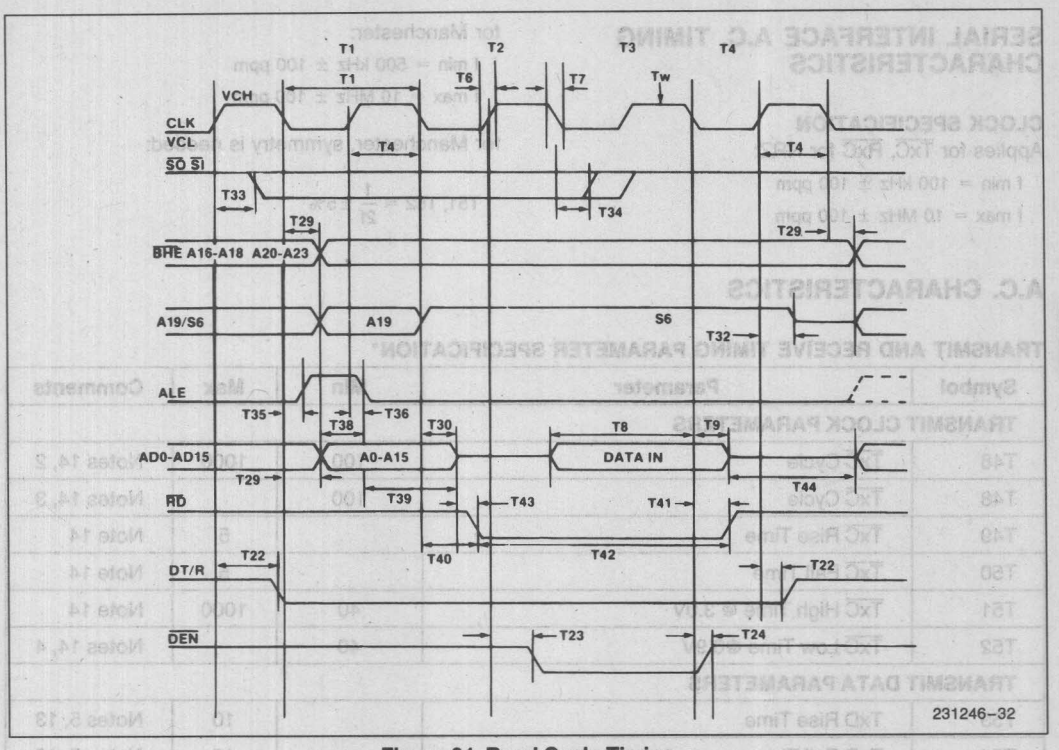


Figure 31. Read Cycle Timing

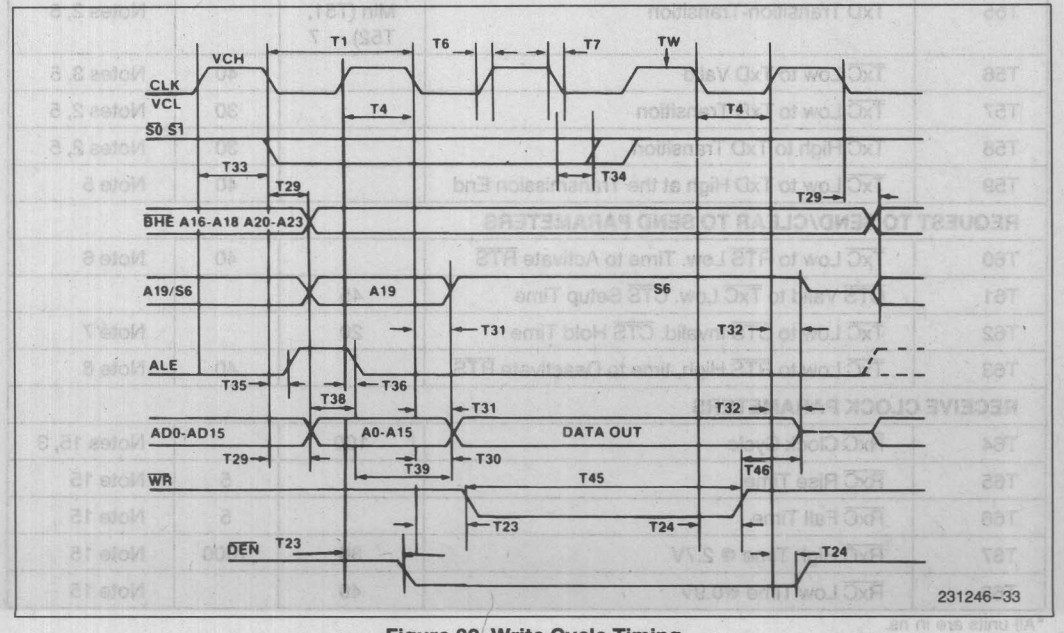


Figure 32. Write Cycle Timing

SERIAL INTERFACE A.C. TIMING CHARACTERISTICS

CLOCK SPECIFICATION

Applies for $\overline{TxC}$, $\overline{RxC}$ for NRZ:

f min = 100 kHz ± 100 ppm

f max = 10 MHz ± 100 ppm

for Manchester:

f min = 500 kHz ± 100 ppm

f max = 10 MHz ± 100 ppm

for Manchester, symmetry is needed:

$$T51, T52 = \frac{1}{2f} \pm 5\%$$

A.C. CHARACTERISTICS

TRANSMIT AND RECEIVE TIMING PARAMETER SPECIFICATION*

Symbol	Parameter	Min	Max	Comments
TRANSMIT CLOCK PARAMETERS				
T48	$\overline{TxC}$ Cycle	100	1000	Notes 14, 2
T48	$\overline{TxC}$ Cycle	100		Notes 14, 3
T49	$\overline{TxC}$ Rise Time		5	Note 14
T50	$\overline{TxC}$ Fall Time		5	Note 14
T51	$\overline{TxC}$ High Time @ 3.0V	40	1000	Note 14
T52	$\overline{TxC}$ Low Time @0.9V	40		Notes 14, 4
TRANSMIT DATA PARAMETERS				
T53	TxD Rise Time		10	Notes 5, 13
T54	TxD Fall Time		10	Notes 5, 13
T55	TxD Transition-Transition	Min (T51, T52) − 7		Notes 2, 5
T56	$\overline{TxC}$ Low to TxD Valid		40	Notes 3, 5
T57	$\overline{TxC}$ Low to TxD Transition		30	Notes 2, 5
T58	$\overline{TxC}$ High to TxD Transition		30	Notes 2, 5
T59	$\overline{TxC}$ Low to TxD High at the Transmission End		40	Note 5
REQUEST TO SEND/CLEAR TO SEND PARAMETERS				
T60	$\overline{TxC}$ Low to $\overline{RTS}$ Low. Time to Activate $\overline{RTS}$		40	Note 6
T61	$\overline{CTS}$ Valid to $\overline{TxC}$ Low. $\overline{CTS}$ Setup Time	45		
T62	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid. $\overline{CTS}$ Hold Time	20		Note 7
T63	$\overline{TxC}$ Low to $\overline{RTS}$ High, time to Deactivate $\overline{RTS}$		40	Note 6
RECEIVE CLOCK PARAMETERS				
T64	$\overline{RxC}$ Clock Cycle	100		Notes 15, 3
T65	$\overline{RxC}$ Rise Time		5	Note 15
T66	$\overline{RxC}$ Fall Time		5	Note 15
T67	$\overline{RxC}$ High Time @ 2.7V	36	1000	Note 15
T68	$\overline{RxC}$ Low Time @0.9V	40		Note 15

*All units are in ns.

A.C. CHARACTERISTICS (Continued)

TRANSMIT AND RECEIVE TIMING PARAMETER SPECIFICATION* (Continued)

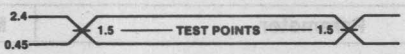
Symbol	Parameter	Min	Max	Comments
RECEIVE DATA PARAMETERS				
T69	RxD Setup Time	30		Note 1
T70	RxD Hold Time	30		Note 1
T71	RxD Rise Time		10	Note 1
T72	RxD Fall Time		10	Note 1
CARRIER SENSE/COLLISION DETECT PARAMETERS				
T73	$\overline{CDT}$ Valid to $\overline{TxC}$ High Ext. Collision Detect Setup Time	30		Note 12
T74	$\overline{TxC}$ High to $\overline{CDT}$ Inactive. $\overline{CDT}$ Hold Time	20		Note 12
T75	$\overline{CDT}$ Low to Jamming Start			Note 8
T76	$\overline{CRS}$ Valid to $\overline{TxC}$ High Ext. Carrier Sense Setup Time	30		Note 12
T77	$\overline{TxC}$ High to $\overline{CRS}$ Inactive. $\overline{CRS}$ Hold Time	20		Note 12
T78	$\overline{CRS}$ Low to Jamming Start			Note 9
T79	Jamming Period			Note 10
T80	$\overline{CRS}$ Inactive Setup Time to $\overline{RxC}$ High End of Receive Frame	60		
T81	$\overline{CRS}$ Active Hold Time from $\overline{RxC}$ High	3		
INTERFRAME SPACING PARAMETER				
T82	Inter Frame Delay			Note 11

*All units are in ns.

NOTES:
1. TTL levels
2. Manchester only
3. NRZ only
4. Manchester requires 50% duty cycle
5. 1 TTL load + 50 pF
6. 1 TTL load + 100 pF
7. Abnormal end of transmission. $\overline{CTS}$ expires before $\overline{RTS}$
8. Programmable value:
   T75 = NCDF × T48 + (12.5 to 23.5) × T48 if collision occurs after preamble
   NCDF—The collision detection filter configuration value
9. Programmable value:
   T78 = NCSF × T48 + (12.5 to 23.5) × T48
   NCSF—The carrier sense filter configuration value
   TBD is a function of internal/external carrier sense bit
10. T79 = 32 × T48
11. Programmable value:
    T82 = NIFS × T48
    NIFS—the IFS configuration value
*12. To guarantee recognition on the next clock
13. Applies to TTL levels
14. 82C501 compatible levels, see Figure 34
15. 82C501 compatible levels, see Figure 35

A.C. TIMING CHARACTERISTICS

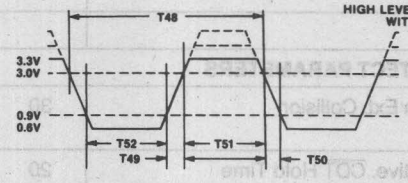
Input and Output Waveforms for AC Tests



231246-34

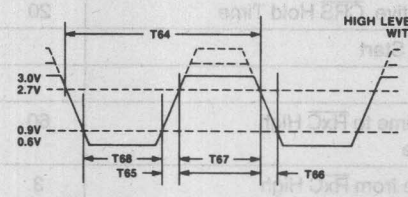
AC testing inputs are driven at 2.4V for a Logic 1 and 0.45 for a Logic 0. Timing measurements are made at 1.5V for both a Logic 1 and 0.

Figure 33. TTL Input/Output Voltage Levels for Timing Measurements



231246-35

Figure 34. $\overline{TxC}$ Input Voltage Levels for Timing Measurements



231246-36

Figure 35. $\overline{RxC}$ Input Voltage Levels for Timing Measurements

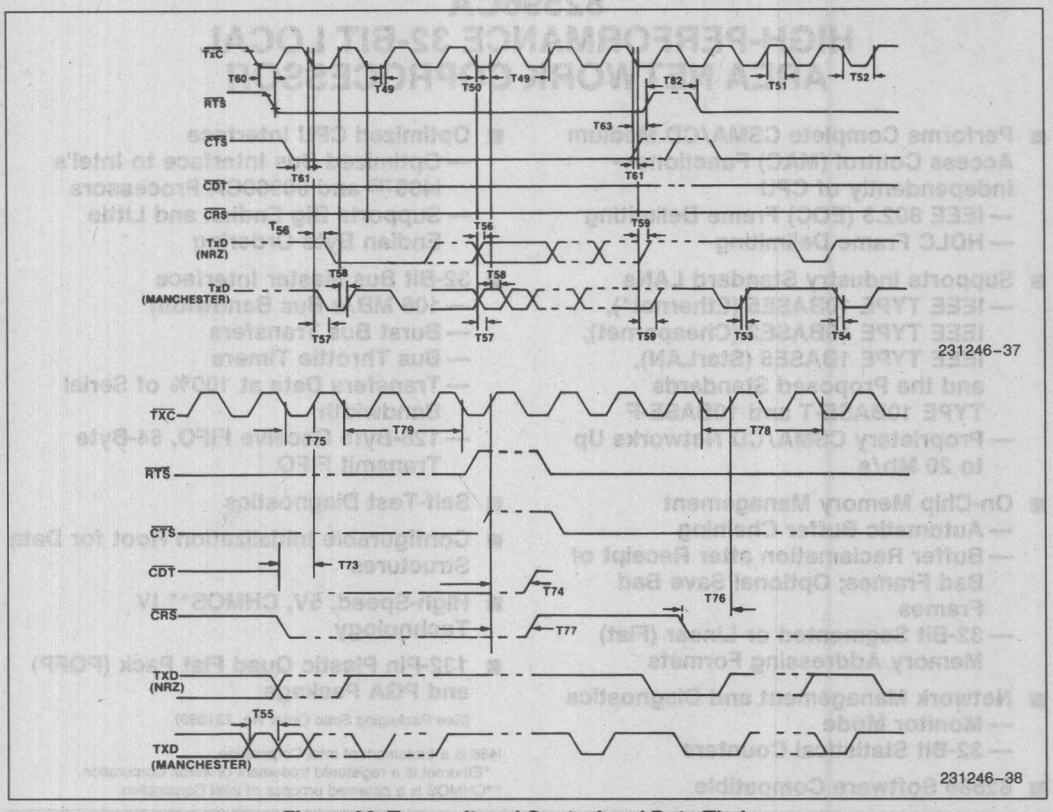


Figure 36. Transmit and Control and Data Timing

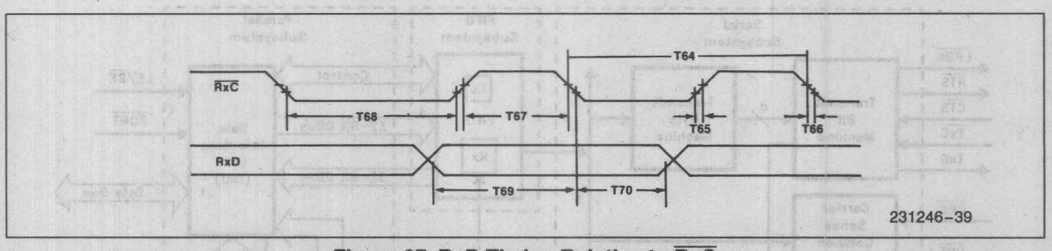


Figure 37. RxD Timing Relative to $\overline{RxC}$

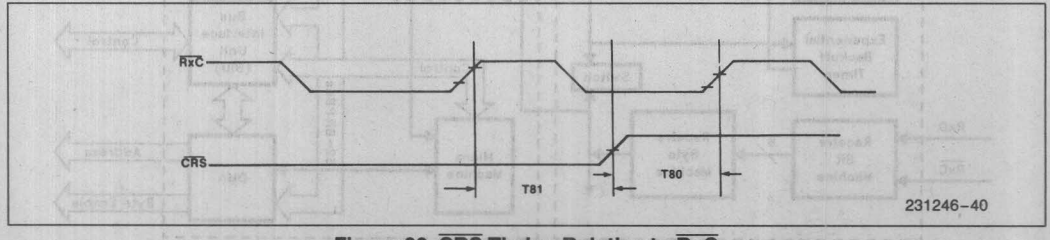


Figure 38. $\overline{CRS}$ Timing Relative to $\overline{RxC}$

82596CA
HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

- Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU
  - IEEE 802.3 (EOC) Frame Delimiting
  - HDLC Frame Delimiting
- Supports Industry Standard LANs
  - IEEE TYPE 10BASE5 (Ethernet*), IEEE TYPE 10BASE2 (Cheapernet), IEEE TYPE 1BASE5 (StarLAN), and the Proposed Standards TYPE 10BASE-T and 10BASE-F
  - Proprietary CSMA/CD Networks Up to 20 Mb/s
- On-Chip Memory Management
  - Automatic Buffer Chaining
  - Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames
  - 32-Bit Segmented or Linear (Flat) Memory Addressing Formats
- Network Management and Diagnostics
  - Monitor Mode
  - 32-Bit Statistical Counters
- 82586 Software Compatible
- Optimized CPU Interface
  - Optimized Bus Interface to Intel's i486™ and 80960CA Processors
  - Supports Big Endian and Little Endian Byte Ordering
- 32-Bit Bus Master Interface
  - 106 MB/s Bus Bandwidth
  - Burst Bus Transfers
  - Bus Throttle Timers
  - Transfers Data at 100% of Serial Bandwidth
  - 128-Byte Receive FIFO, 64-Byte Transmit FIFO
- Self-Test Diagnostics
- Configurable Initialization Root for Data Structures
- High-Speed, 5V, CHMOS** IV Technology
- 132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package

  (See Packaging Spec Order No. 231369)

i486 is a trademark of Intel Corporation.
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.

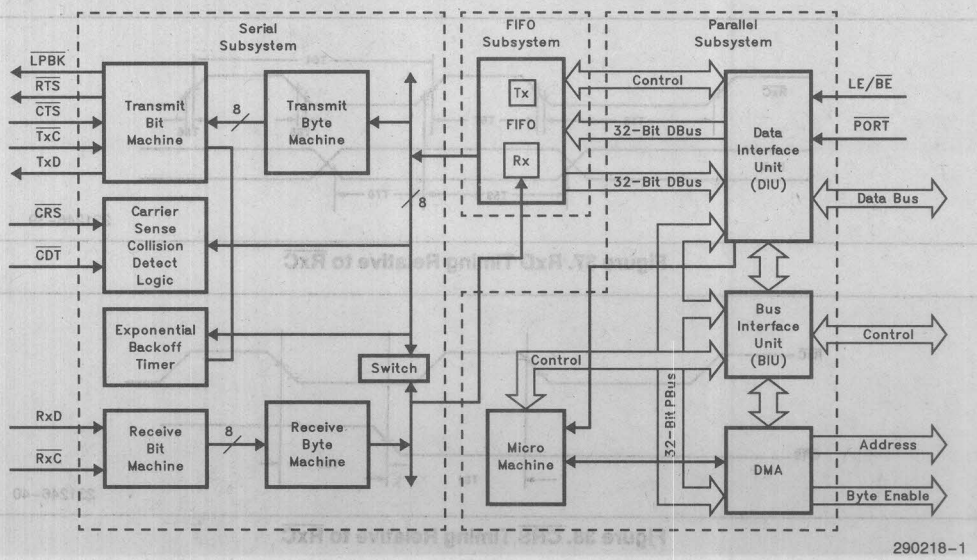


Figure 1. 82596CA Block Diagram

October 1990
Order Number: 290218-003

82596CA High-Performance 32-Bit Local Area Network Coprocessor

CONTENTS

PAGE

INTRODUCTION 1-40

PIN DESCRIPTIONS 1-44

82596 AND HOST CPU INTERACTION 1-48

82596 BUS INTERFACE 1-48

82596 MEMORY ADDRESSING 1-48

82596 SYSTEM MEMORY STRUCTURE 1-50

TRANSMIT AND RECEIVE MEMORY STRUCTURES 1-51

TRANSMITTING FRAMES 1-54

RECEIVING FRAMES 1-55

82596 NETWORK MANAGEMENT AND DIAGNOSTICS 1-55

NETWORK PLANNING AND MAINTENANCE 1-57

STATION DIAGNOSTICS AND SELF-TEST 1-58

82586 SOFTWARE COMPATIBILITY ... 1-58

INITIALIZING THE 82596 1-58

SYSTEM CONFIGURATION POINTER (SCP) 1-58

Writing the Sysbus 1-59

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP) 1-60

INITIALIZATION PROCESS 1-60

CONTROLLING THE 82596CA 1-61

82596 CPU ACCESS INTERFACE (PORT) 1-61

MEMORY ADDRESSING FORMATS 1-62

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING 1-62

COMMAND UNIT (CU) 1-62

CONTENTS

PAGE

RECEIVE UNIT (RU) 1-63

SYSTEM CONTROL BLOCK (SCB) 1-63

SCB OFFSET ADDRESSES 1-66

CBL Offset (Address) 1-66

RFA Offset (Address) 1-66

SCB STATISTICAL COUNTERS 1-67

Statistical Counter Operation 1-67

ACTION COMMANDS AND OPERATING MODES 1-68

NOP 1-69

Individual Address Setup 1-69

Configure 1-70

Multicast-Setup 1-76

Transmit 1-77

Jamming Rules 1-79

TDR 1-80

Dump 1-82

Diagnose 1-85

RECEIVE FRAME DESCRIPTOR 1-86

Simplified Memory Structure 1-86

Flexible Memory Structure 1-87

Receive Buffer Descriptor (RBD) 1-88

ELECTRICAL AND TIMING CHARACTERISTICS 1-93

DC Characteristics 1-93

AC Characteristics 1-94

82596CA Input/Output System Timings 1-94

Transmit/Receive Clock Parameters 1-96

82596CA BUS Operation 1-99

System Interface AC Timing Characteristics 1-100

Input Waveforms 1-101

Serial AC Timing Characteristics 1-103

OUTLINE DIAGRAM 1-105

REVISION HISTORY 1-109

formance and efficiency. The 82596CA bus interfaces is optimized for Intel's i486™, 80960CA, and 80960KB processors.

The 82596CA implements all IEEE 802.3 Medium Access Control and channel interface functions, these include framing, preamble generation and stripping, source address generation, destination address checking, short-frame detection, and automatic length-field handling. Data rates up to 20 Mb/s are supported.

The 82596CA provides a powerful host system interface. It manages memory structures automatically, with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels, this allows autonomous transfer of data blocks (buffers and frames) and relieves the CPU of byte transfer overhead. Buffers containing errored or collided frames can be automatically recovered without CPU intervention. The 82596CA provides an upgrade path for existing 82586 software drivers by providing an 82586-software-compatible mode that supports the current 82586 memory structure. The 82586CA also has a Flexible memory structure and a Simplified memory structure. The 82596CA can address up to 4 gigabytes of memory. The 82596CA supports Little Endian and Big Endian byte ordering.

The 82596CA bus interface can achieve a burst transfer rate of 106 MB/s at 33 MHz. The bus interface is optimized for Intel's i486™, 80960CA, and 80960KB processors.

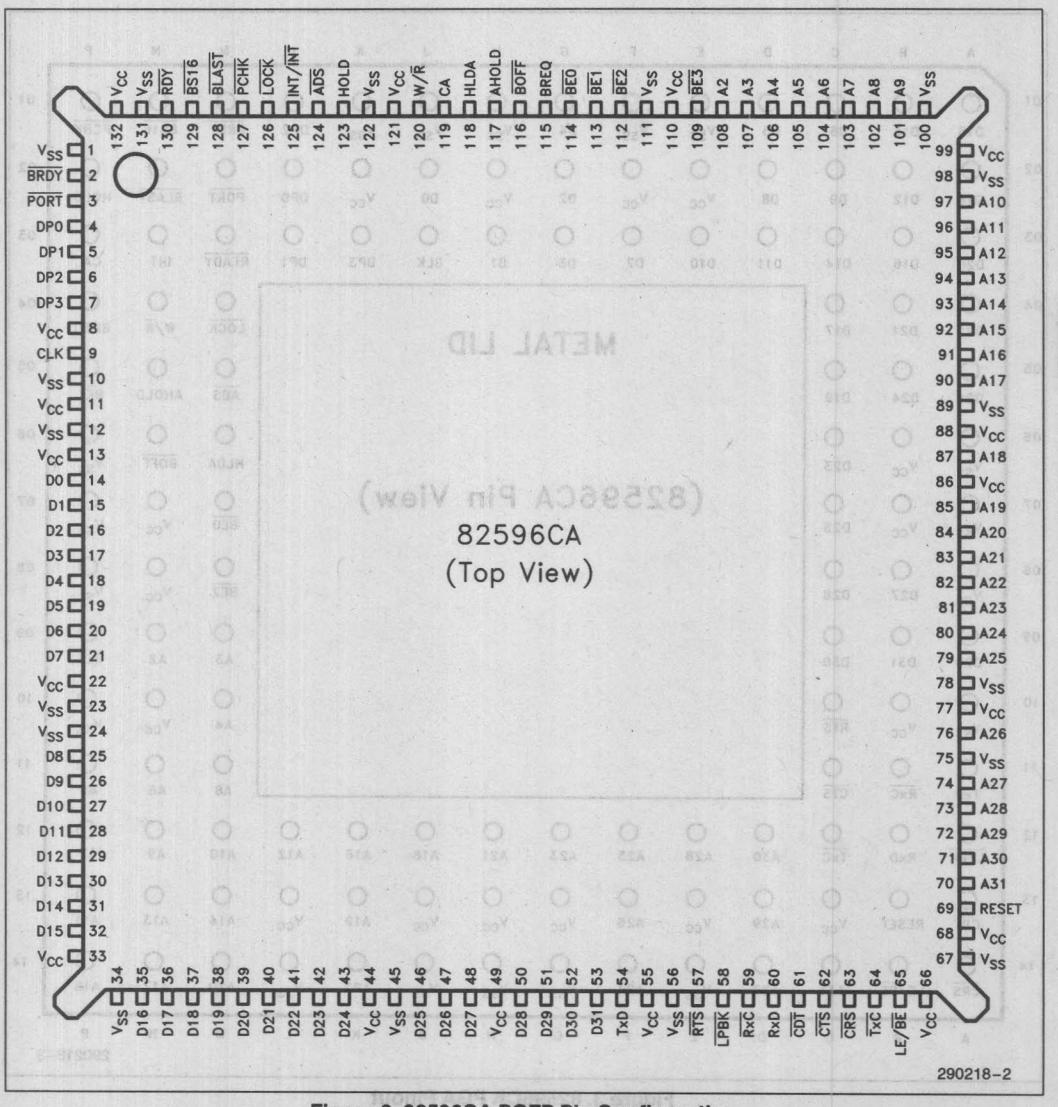


Figure 2. 82596CA PQFP Pin Configuration

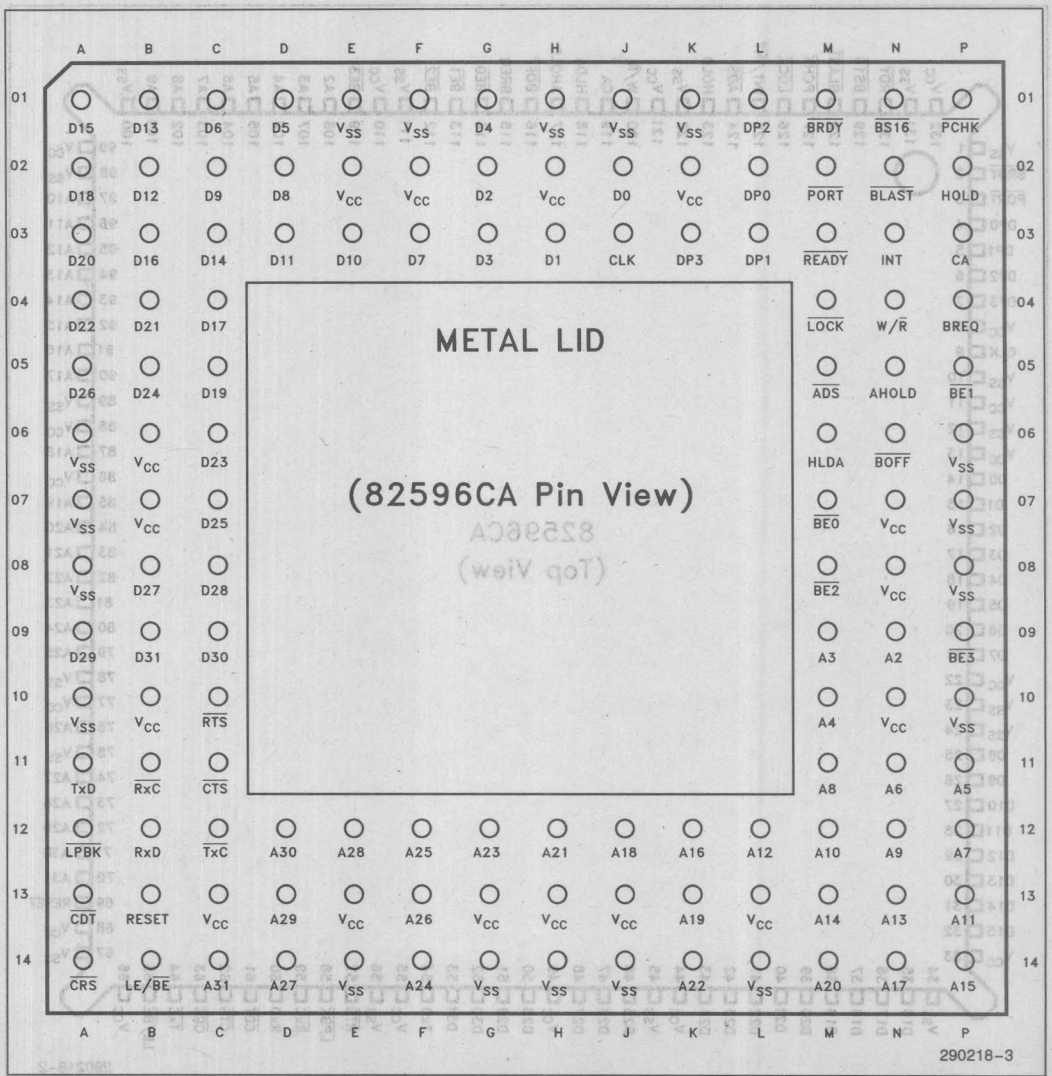


Figure 3. 82596CA PGA Pinout

82596CA PGA Cross Reference by Pin Name

Address		Data		Control		Serial Interface		V_{CC}	V_{SS}
Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Pin No.	Pin No.
A2	N9	D0	J2	$\overline{ADS}$	M5	$\overline{CDT}$	A13	B6	A6
A3	M9	D1	H3	AHOLD	N5	$\overline{CRS}$	A14	B7	A7
A4	M10	D2	G2	$\overline{BE0}$	M7	$\overline{CTS}$	C11	B10	A8
A5	P11	D3	G3	$\overline{BE1}$	P5	$\overline{LPBK}$	A12	C13	A10
A6	N11	D4	G1	$\overline{BE2}$	M8	$\overline{RTS}$	C10	E2	E1
A7	P12	D5	D1	$\overline{BE3}$	P9	$\overline{RxC}$	B11	E13	E14
A8	M11	D6	C1	$\overline{BLAST}$	N2	$\overline{RxD}$	B12	F2	F1
A9	N12	D7	F3	$\overline{BOFF}$	N6	$\overline{TxC}$	C12	G13	G14
A10	M12	D8	D2	$\overline{BRDY}$	M1	TxD	A11	H2	H1
A11	P13	D9	C2	BREQ	P4			H13	H14
A12	L12	D10	E3	$\overline{BS16}$	N1			J13	J1
A13	N13	D11	D3	CA	P3			K2	J14
A14	M13	D12	B2	CLK	J3			L13	K1
A15	P14	D13	B1	DP0	L2			N7	L14
A16	K12	D14	C3	DP1	L3			N8	P6
A17	N14	D15	A1	DP2	L1			N10	P7
A18	J12	D16	B3	DP3	K3				P8
A19	K13	D17	C4	HLDA	M6				P10
A20	M14	D18	A2	HOLD	P2				
A21	H12	D19	C5	INT/$\overline{INT}$	N3				
A22	K14	D20	A3	LE/$\overline{BE}$	B14				
A23	G12	D21	B4	$\overline{LOCK}$	M4				
A24	F14	D22	A4	$\overline{PCHK}$	P1				
A25	F12	D23	C6	$\overline{PORT}$	M2				
A26	F13	D24	B5	$\overline{READY}$	M3				
A27	D14	D25	C7	RESET	B13				
A28	E12	D26	A5	W/$\overline{R}$	N4				
A29	D13	D27	B8						
A30	D12	D28	C8						
A31	C14	D29	A9						
		D30	C9						
		D31	B9						

PIN DESCRIPTIONS

<table>
<tr><th>Symbol</th><th>PQFP Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>CLK</td><td>9</td><td>I</td><td>CLOCK. The system clock input provides the fundamental timing for the 82596. It is a 1X CLK input used to generate the 82596 clock and requires TTL levels. All external timing parameters are specified in reference to the rising edge of CLK.</td></tr>
<tr><td>D0–D31</td><td>14–53</td><td>I/O</td><td>DATA BUS. The 32 Data Bus lines are bidirectional, tri-state lines that provide the general purpose data path between the 82596 and memory. With the 82596 the bus can be either 16 or 32 bits wide; this is determined by the $\overline{BS16}$ signal. The 82596 always drives all 32 data lines during Write operations, even with a 16-bit bus. D31– D0 are floated after a Reset or when the bus is not acquired.
These lines are inputs during a CPU Port access; in this mode the CPU writes the next address to the 82596 through the data lines. During $\overline{PORT}$ commands (Relocatable SCP, Self-Test, Reset and Dump) the address must be aligned to a 16-byte boundary. This frees the D_3–D_0 lines so they can be used to distinguish the commands. The following is a summary of the decoding data.
<table>
<tr><th>D0</th><th>D1</th><th>D2</th><th>D3</th><th>D31–D4</th><th>Function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0000</td><td>Reset</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ADDR</td><td>Relocatable SCP</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>ADDR</td><td>Self-Test</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>ADDR</td><td>Dump Command</td></tr>
</table></td></tr>
<tr><td>DP0–DP3</td><td>4–7</td><td>I/O</td><td>DATA PARITY. These are tri-stated data parity pins. There is one parity line for each byte of the data bus. The 82596 drives them with even-parity information during write operations having the same timing as data writes. Likewise, even-parity information, with the same timing as read information, must be driven back to the 82596 over these pins to ensure that the correct parity check status is indicated by the 82596.</td></tr>
<tr><td>$\overline{PCHK}$</td><td>127</td><td>O</td><td>PARITY CHECK. This pin is driven high one clock after $\overline{RDY}$ to inform Read operations of the parity status of data sampled at the end of the previous clock cycle. When driven low it indicates that incorrect parity data has been sampled. It only checks the parity status of enabled bytes, which are indicated by the Byte Enable and Bus Size signals. $\overline{PCHK}$ is only valid for one clock time after data read is returned to the 82596; i.e., it is inactive (high) at all other times.</td></tr>
<tr><td>A31-A2</td><td>7008</td><td>O</td><td>ADDRESS LINES. These 30 tri-stated Address lines output the address bits required for memory operation. These lines are floated after a Reset or when the bus is not acquired.</td></tr>
<tr><td>$\overline{BE3}$–$\overline{BE0}$</td><td>109–114</td><td>O</td><td>BYTE ENABLE. These tri-stated signals are used to indicate which bytes are involved with the current memory access. The number of Byte Enable signals asserted indicates the physical size of the data being transferred (1, 2, 3, or 4 bytes).
• $\overline{BE0}$ indicates D7–D0
• $\overline{BE1}$ indicates D15–D8
• $\overline{BE2}$ indicates D23–D16
• $\overline{BE3}$ indicates D31–D24
These lines are floated after a Reset or when the bus is not acquired.</td></tr>
<tr><td>W/$\overline{R}$</td><td>120</td><td>O</td><td>WRITE/READ. This dual function pin is used to distinguish Write and Read cycles. This line is floated after a Reset or when the bus is not acquired.</td></tr>
</table>

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{ADS}$	124	O	**ADDRESS STATUS.** The 82596 uses this tri-state pin to indicate to indicate that a valid bus cycle has begun and that A31–A2, $\overline{BE3}$–$\overline{BE0}$, and W/$\overline{R}$ are being driven. It is asserted during t1 bus states. This line is floated after a Reset or when the bus is not acquired.
$\overline{RDY}$	130	I	**READY.** Active low. This signal is the acknowledgment from addressed memory that the transfer cycle can be completed. When high, it causes wait states to be inserted. It is ignored at the end of the first clock of the bus cycle's data cycle. This active-low signal does not have an internal pull-up resistor. This signal must meet the setup and hold times to operate correctly.
$\overline{BRDY}$	2	I	**BURST READY.** Active low. Burst Ready, like $\overline{RDY}$, indicates that the external system has presented valid data on the data pins in response to a Read, or that the external system has accepted the 82596 data in response to a Write request. Also, like $\overline{RDY}$, this signal is ignored at the end of the first clock in a bus cycle. If the 82596 can still receive data from the previous cycle, ADS will not be asserted in the next clock cycle; however, Address and Byte Enable will change to reflect the next data item expected by the 82596. $\overline{BRDY}$ will be sampled during each succeeding clock and if active, the data on the pins will be strobed to the 82596 or to external memory (read/write). $\overline{BRDY}$ operates exactly like READY during the last data cycle of a burst sequence and during nonburstable cycles.
$\overline{BLAST}$	128	O	**BURST LAST.** A signal (active low) on this tri-state pin indicates that the burst cycle is finished and when $\overline{BRDY}$ is next returned it will be treated as a normal ready; i.e., another set of addresses will be driven with $\overline{ADS}$ or the bus will go idle. $\overline{BLAST}$ is not asserted if the bus is not acquired.
AHOLD	117	I	**ADDRESS HOLD.** This hold signal is active high, it allows another bus master to access the 82596 address bus. In a system where an 82596 and an i486 processor share the local bus, AHOLD allows the cache controller to make a cache invalidation cycle while the 82596 holds the address lines. In response to a signal on this pin, the 82596 immediately (i.e. during the next clock) stops driving the entire address bus (A31–A2); the rest of the bus can remain active. For example, data can be returned for a previously specified bus cycle during Address Hold. The 82596 will not begin another bus cycle while AHOLD is active.
$\overline{BOFF}$	116	I	**BACKOFF.** This signal is active low, it informs the 82596 that another bus master requires access to the bus before the 82596 bus cycle completes. The 82596 immediately (i.e. during the next clock) floats its bus. Any data returned to the 82596 while $\overline{BOFF}$ is asserted is ignored. $\overline{BOFF}$ has higher priority than $\overline{RDY}$ or $\overline{BRDY}$; if two such signals are returned in the same clock period, $\overline{BOFF}$ is given preference. The 82596 remains in Hold until $\overline{BOFF}$ goes high, then the 82596 resumes its bus cycle by driving out the address and status, and asserting $\overline{ADS}$.
$\overline{LOCK}$	126	O	**LOCK.** This tri-state pin is used to distinguish locked and unlocked bus cycles. $\overline{LOCK}$ generates a semaphore handshake to the CPU. $\overline{LOCK}$ can be active for several memory cycles, it goes active during the first locked memory cycle (t1) and goes inactive at the last locked cycle (t2). This line is floated after a Reset or when the bus is not acquired. $\overline{LOCK}$ can be disabled via the sysbus byte in software.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{BS16}$	129	I	**BUS SIZE.** This signal allows the 82596CA to work with either 16- or 32-bit bytes. Inserting $\overline{BS16}$ low causes the 82596 to perform two 16-bit memory accesses when transferring 32-bit data. In little endian mode the D15–D0 lines are driven when $\overline{BS16}$ is inserted, in Big Endian mode the D31–D16 lines are driven.
HOLD	123	O	**HOLD.** The HOLD signal is active high, the 82596 uses it to request local bus mastership. In normal operation HOLD goes inactive before HLDA. The 82596 can be forced off the bus by deasserting HLDA or if the bus throttle timers expire.
HLDA	118	I	**HOLD ACKNOWLEDGE.** The HLDA signal is active high, it indicates that bus mastership has been given to the 82596. HLDA is internally synchronized; after HOLD is detected low, the CPU drives HLDA low. **NOTE** *Do not connect HLDA to V_{CC}—it will cause a deadlock.* A user wanting to give the 82596 permanent access to the bus should connect HLDA to HOLD. If HLDA goes inactive before HOLD, the 82596 will release the bus (by deasserting HOLD) within a maximum of within a specified number of bus cycles as specified in the 82596 User's Manual.
BREQ	115	I	**BUS REQUEST.** This signal, when configured to an externally activated mode, is used to trigger the bus throttle timers.
$\overline{PORT}$	3	I	**PORT.** When this signal is received, the 82596 latches the data on the data bus into an internal 32-bit register. When the CPU is asserting this signal it can write into the 82596 (via the data bus). This pin must be activated twice during all CPU Port access commands.
RESET	69	I	**RESET.** This active high, internally synchronized signal causes the 82596 to terminate current activity. The signal must be high for at least five system clock cycles. After five system clock cycles and four $\overline{TxC}$ clock cycles the 82596 will execute a Reset when it receives a high RESET signal. When RESET returns to low the 82596 waits for the first CA signal and then begins the initialization sequence.
LE/$\overline{BE}$	65	I	**LITTLE ENDIAN/BIG ENDIAN.** This dual-function pin is used to select byte ordering. When LE/$\overline{BE}$ is high, little endian byte ordering is used; when low, big endian byte ordering is used for data in frames (bytes) and for control (SCB, RFD, CBL, etc).
CA	119	I	**CHANNEL ATTENTION.** The CPU uses this pin to force the 82596 to begin executing memory resident Command blocks. The CA signal is internally synchronized. The signal must be high for at least one system clock. It is latched internally on the high to low edge and then detected by the 82596. The first CA after a Reset forces the 82596 into the initialization sequence beginning at location 00FFFFF6h or an SCP address written to the 82596 using CPU Port access. All subsequent CA signals cause the 82596 to begin executing new command sequences from the SCB.
INT/$\overline{INT}$	125	O	**INTERRUPT.** A high signal on this pin notifies the CPU that the 82596 is requesting an interrupt. This signal is an edge triggered interrupt signal, and can be configured to be active high or low.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
V_{CC}	18 Pins		**POWER.** +5 V ±10%.
V_{SS}	18 Pins		**GROUND.** 0 V.
TxD	54	O	**TRANSMIT DATA.** This pin transmits data to the serial link. It is high when not transmitting.
$\overline{TxC}$	64	I	**TRANSMIT CLOCK.** This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the high to low clock transition. For Manchester encoding, the transmitted bit center is aligned with the low to high transition. Transmit clock must always be running for proper device operation.
$\overline{LPBK}$	58	O	**LOOPBACK.** This TTL-level control signal enables the loopback mode. In this mode serial data on the TxD input is routed through the 82C501 internal circuits and back to the RxD output without driving the transceiver cable. To enable this signal, both internal and external loopback need to be set with the Configure command.
RxD	60	I	**RECEIVE DATA.** This pin receives NRZ serial data only. It must be high when not receiving.
$\overline{RxC}$	59	I	**RECEIVE CLOCK.** This signal provides timing information to the internal shifting logic. For NRZ data the state of the RxD pin is sampled on the high to low transition of the clock.
$\overline{RTS}$	57	O	**REQUEST TO SEND.** When this signal is low the 82596 informs the external interface that it has data to transmit. It is forced high after a Reset or when transmission is stopped.
$\overline{CTS}$	62	I	**CLEAR TO SEND.** An active-low signal that enables the 82596 to send data. It is normally used as an interface handshake to $\overline{RTS}$. Asserting $\overline{CTS}$ high stops transmission. $\overline{CTS}$ is internally synchronized. If $\overline{CTS}$ goes inactive, meeting the setup time to the $\overline{TxC}$ negative edge, the transmission will stop and $\overline{RTS}$ will go inactive within, at most, two $\overline{TxC}$ cycles.
$\overline{CRS}$	63	I	**CARRIER SENSE.** This signal is active low, it is used to notify the 82596 that traffic is on the serial link. It is only used if the 82596 is configured for external Carrier Sense. In this configuration external circuitry is required for detecting traffic on the serial link. $\overline{CRS}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CRSF = 0).
$\overline{CDT}$	61	I	**COLLISION DETECT.** This active-low signal informs the 82596 that a collision has occurred. It is only used if the 82596 is configured for external Collision Detect. External circuitry is required for collision detection. $\overline{CDT}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CDTF = 0).

82596 AND HOST CPU INTERACTION

The 82596CA and the host CPU communicate through shared memory. Because of its on-chip DMA capability, the 82596 can make data block transfers (buffers and frames) independently of the CPU; this greatly reduces the CPU byte transfer overhead.

The 82596 is a multitasking coprocessor that comprises two independent logical units—the Command Unit (CU) and the Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The independence of the CU and RU enables the 82596 to engage in both activities simultaneously—the CU can fetch and execute commands from memory while the RU is storing received frames in memory. The CPU is only involved with this process after the CU has executed a sequence of commands or the RU has finished storing a sequence of frames.

The CPU and the 82596 use the hardware signals Interrupt (INT) and Channel Attention (CA) to initiate communication with the System Control Block (SCB), see Figure 4. The 82596 uses INT to alert the CPU of a change in the contents of the SCB, the CPU uses CA to alert the 82596.

The 82596 has a CPU Port Access state that allows the CPU to execute certain functions without accessing memory. The 82596 $\overline{\text{PORT}}$ pin and data bus pins are used to enable this feature. The CPU can directly activate four operations when the 82596 is in this state.

- Write an alternative System Configuration Pointer (SCP). This can be used when the 82596 cannot use the default SCP address space.
- Write a different Dump Command Pointer and execute Dump. This can be used for troubleshooting No Response problems.
- The CPU can reset the 82596 via software without disturbing the rest of the system.
- A self-test can be used for board testing; the 82596 will execute a self-test and write the results to memory.

82596 BUS INTERFACE

The 82596CA has bus interface timings and pin definitions that are compatible with Intel's 32-bit i486 microprocessor. This eliminates the need for additional bus interface logic. Operating at 33 MHz, the 82596's bus bandwidth can be as high as 106 MB/s. Since Ethernet only requires 1.25 MB/s, this leaves a considerable amount of bandwidth for the CPU. The 82596 also has a bus throttle to regulate its use of the bus. Two timers can be programmed through the SCB: one controls the maximum time the 82596 can remain on the bus, the other controls the time the 82596 must stay off the bus (see Figure 5). The bus throttle can be programmed to trigger internally with HLDA or externally with BREQ. These timers can restrict the 82596 HOLD activation time and improve bus utilization.

82596 MEMORY ADDRESSING

The 82596 has a 32-bit memory address range, which allows addressing up to four gigabytes of memory. The 82596 has three memory addressing modes (see Table 1).

- **82586 Mode.** The 82596 has a 24-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 24-bit address space.
- **32-Bit Segmented Mode.** The 82596 has a 32-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-KB memory segment. Transmit and Receive buffers can reside in a 32-bit address space.
- **Linear Mode.** The 82596 has a 32-bit memory address range. Any memory structure can reside anywhere within the 32-bit memory address range.

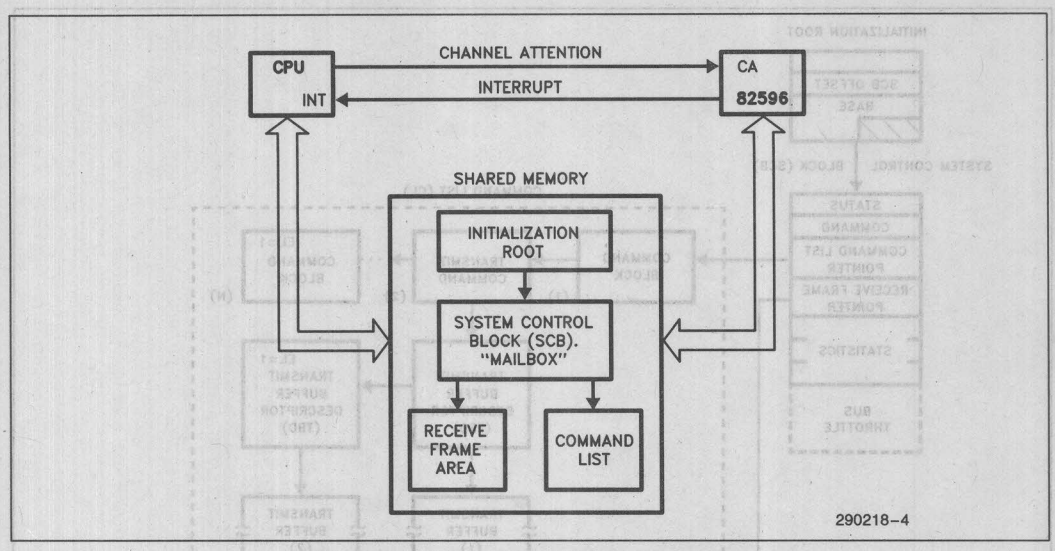


Figure 4. 82596 and Host CPU Intervention

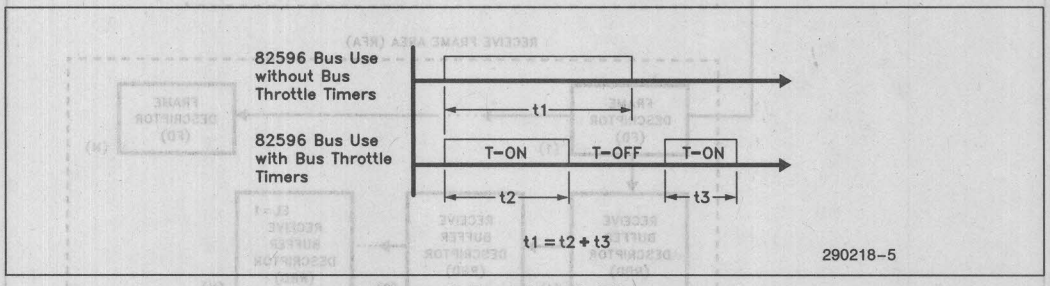


Figure 5. Bus Throttle Timers

Table 1. 82596 Memory Addressing Formats

Pointer or Offset	Operation Mode		
	82586	32-Bit Segmented	Linear
ISCP Address	24-Bit Linear	32-Bit Linear	32-Bit Linear
SCB Address	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Command Block Pointers	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear
Tx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear

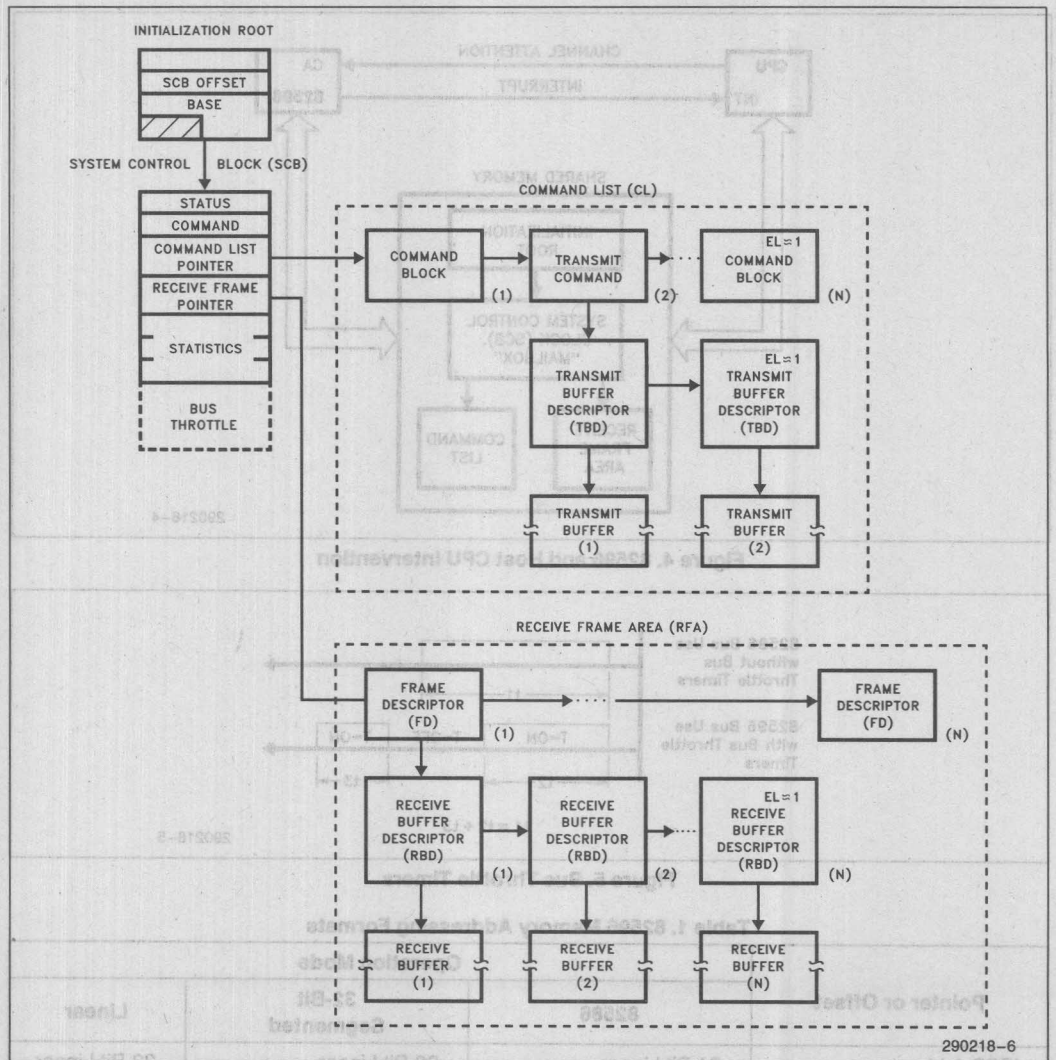


Figure 6. 82596 Shared Memory Structure

82596 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: the Initialization Root, the System Control Block, the Command List, and the Receive Frame Area (see Figure 6).

The Initialization Root is in an established location known to the host CPU and the 82596 (00FFFFF6h). However, the CPU can establish the Initialization Root in another location by using the CPU Port access. This root is accessed during initialization, and points to the System Control Block.

The System Control Block serves as a bidirectional mail drop for the host CPU and the 82596 CU and RU. It is the central point through which the CPU and the 82596 exchange control and status information. The SCB has two areas. The first contains instructions from the CPU to the 82596. These include: control of the CU and RU (Start, Abort, Suspend, and Resume), a pointer to the list of CU commands, a pointer to the Receive Frame Area, a set of Interrupt Acknowledge bits, and the T-ON and T-OFF timers for the bus throttle. The second area contains status information the 82596 is sending to the CPU. Such as, the CU and RU states (Idle, Active

Ready, Suspended, No Receive Resources, etc.), interrupt bits (Command Completed, Frame Received, CU Not Ready, and RU Not Ready), and statistical counters.

The Command List functions as a program for the CU; individual commands are placed in memory units called Command Blocks (CBs). These CBs contain the parameters and status of specific high-level commands called Action Commands; e.g., Transmit or Configure.

Transmit causes the 82596 to transmit a frame. The Transmit CB contains the destination address, the length field, and a pointer to a list of linked buffers holding the frame that is to be constructed from several buffers scattered throughout memory. The Command Unit operates without CPU intervention; the DMA for each buffer, and the prefetching of references to new buffers, is performed in parallel. The CPU is notified only after a transmission is complete.

The Receive Frame Area is a list of Free Frame Descriptors (descriptors not yet used) and a list of user-prepared buffers. Frames arrive at the 82596 unsolicited; the 82596 must always be ready to receive and store them in the Free Frame Area. The Receive Unit fills the buffers when it receives frames, and reformats the Free Buffer List into received-frame structures. The frame structure is, for all practical purposes, identical to the format of the frame to be transmitted. The first Frame descriptor is referenced by the SCB. Unless the 82596 is configured to Save Bad Frames, the frame descriptor, and the associated buffer descriptor, which is wasted when a bad frame is received, are automatically reclaimed and returned to the Free Buffer List.

Receive buffer chaining (storing incoming frames in a linked buffer list) significantly improves memory utilization. Without buffer chaining, the user must allocate consecutive blocks of memory, each capable of containing a maximum frame (for Ethernet, 1518 bytes). Since an average frame is about 200 bytes, this is very inefficient. With buffer chaining, the user can allocate small buffers and the 82596 will only use those that are needed.

Figure 7 A–D illustrates how the 82596 uses the Receive Frame Area. Figure 7A shows an unused Receive Frame Area composed of Free Frame Descriptors and Free Receive Buffers prepared by the user. The SCB points to the first Frame Descriptor of the Frame Descriptor List. Figure 7B shows the same Receive Frame Area after receiving one frame. This first frame occupies two Receive Buffers and one Frame Descriptor—a valid received frame will only occupy one Frame Descriptor. After receiving this frame the 82596 sets the next Free Frame Descriptor RBD pointer to the next Free RBD. Figure 7C shows the RFA after receiving a second frame. In this example the second frame occupies only one Receive Buffer and one RFD. The 82596 again sets the RBD pointer. This process is repeated again in Figure 7D, showing the reception of another frame using one Receive Buffer; in this example there is an extra Frame Descriptor.

TRANSMIT AND RECEIVE MEMORY STRUCTURES

There are three memory structures for reception and transmission. The 82586 memory structure, the Flexible memory structure, and the Simplified memory structure. The 82586 mode is selected by configuring the 82596 during initialization. In this mode all the 82596 memory structures are compatible with the 82586 memory structures.

When the 82596 is not configured to the 82586 mode, the other two memory structures, Simplified and Flexible, are available for transmitting and receiving. These structures can be selected on a frame-by-frame basis by setting the S/F bit in the Transmit Command and the Receive Frame Descriptor (see Figures 29, 30, 41, and 42). The Simplified memory structure offers a simple structure for ease of programming (see Figure 8). All information about a frame is contained in one structure; for example, during reception the RFD and data field are contained in one structure.

The Flexible memory structure (see Figure 9) has a control field that allows the programmer to specify the amount of receive data the RFD will contain for receive operations and the amount of transmit data the Transmit Command Block will contain for transmit operations. For example, when the control field in the RFD is set to 20 bytes during a reception, the first 20 bytes of the data field are stored in the RFD (6 bytes of destination address, 6 bytes of source address, 2 bytes of length field, and 6 bytes of data) and the remainder of the data field is stored in the Receive Data Buffers. This is useful for capturing frame headers when header information is contained in the data field. The header information can then be automatically stored in the RFD partitioned from the Receive Data Buffer.

The control field can also be used for the Transmit Command when the Flexible memory structure is used. The quantity of data field bytes to be transmitted from the Transmit Command Block is specified by the variable control field.

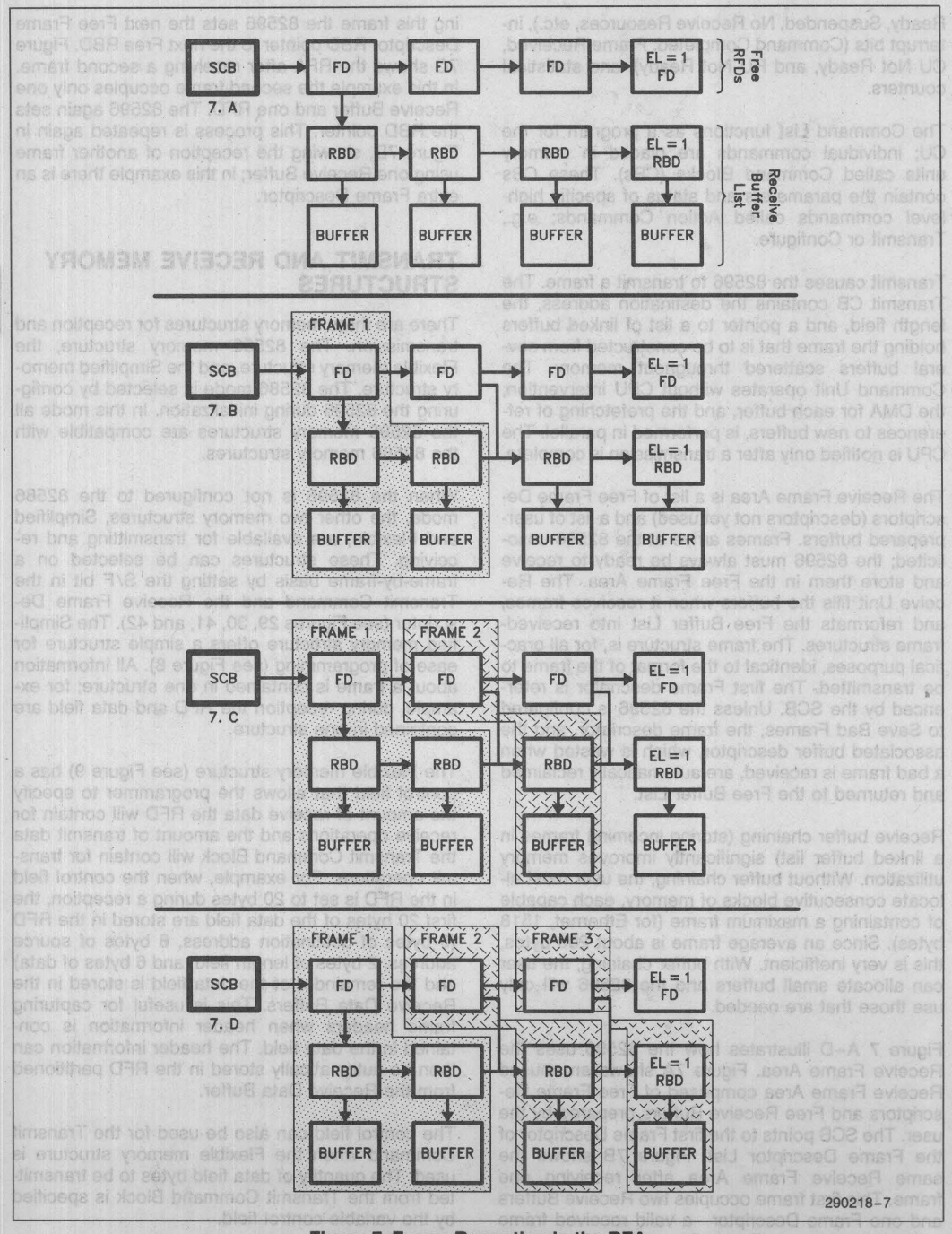


Figure 7. Frame Reception in the RFA

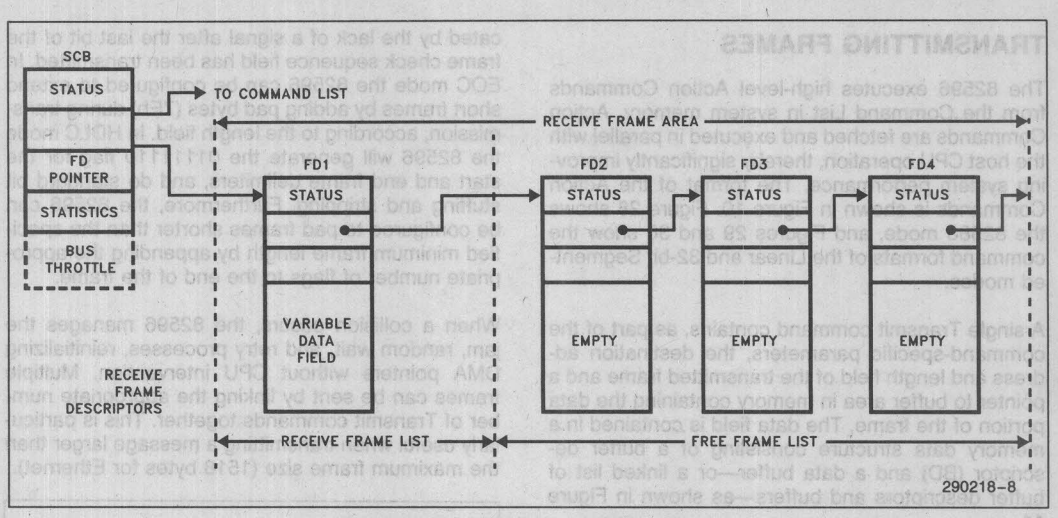


Figure 8. Simplified Memory Structure

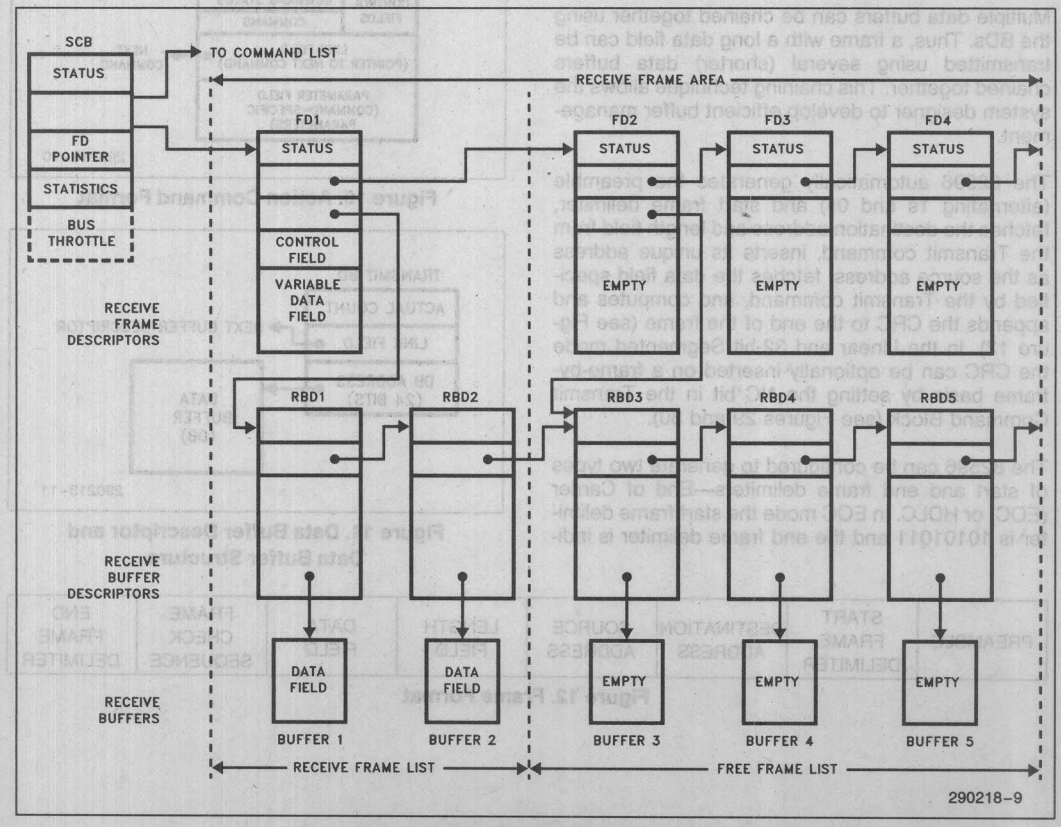


Figure 9. Flexible Memory Structure

pointer to buffer area in memory containing the data portion of the frame. The data field is contained in a memory data structure consisting of a buffer descriptor (BD) and a data buffer—or a linked list of buffer descriptors and buffers—as shown in Figure 11.

Multiple data buffers can be chained together using the BDs. Thus, a frame with a long data field can be transmitted using several (shorter) data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management.

The 82596 automatically generates the preamble (alternating 1s and 0s) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field specified by the Transmit command, and computes and appends the CRC to the end of the frame (see Figure 12). In the Linear and 32-bit Segmented mode the CRC can be optionally inserted on a frame-by-frame basis by setting the NC bit in the Transmit Command Block (see Figures 29 and 30).

The 82596 can be configured to generate two types of start and end frame delimiters—End of Carrier (EOC) or HDLC. In EOC mode the start frame delimiter is 10101011 and the end frame delimiter is indi-

frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message larger than the maximum frame size (1518 bytes for Ethernet).

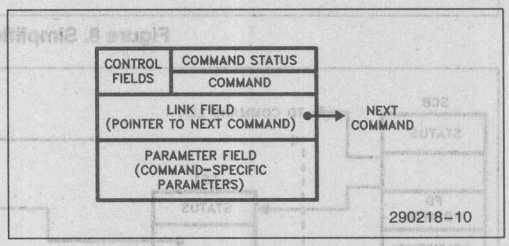


Figure 10. Action Command Format

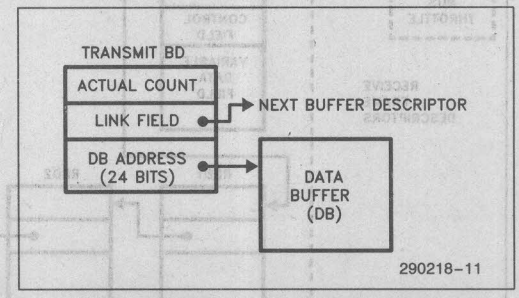


Figure 11. Data Buffer Descriptor and Data Buffer Structure

PREAMBLE	START FRAME DELIMITER	DESTINATION ADDRESS	SOURCE ADDRESS	LENGTH FIELD	DATA FIELD	FRAME CHECK SEQUENCE	END FRAME DELIMITER

Figure 12. Frame Format

RECEIVING FRAMES

To reduce CPU overhead, the 82596 is designed to receive frames without CPU supervision. The host CPU first sets aside an adequate receive buffer space and then enables the 82596 Receive Unit. Once enabled, the RU watches for arriving frames and automatically stores them in the Receive Frame Area (RFA). The RFA contains Receive Frame Descriptors, Receive Buffer Descriptors, and Data Buffers (see Figure 13). The individual Receive Frame Descriptors make up a Receive Descriptor List (RDL) used by the 82596 to store the destination and source addresses, the length field, and the status of each frame received (see Figure 14).

Once enabled, the 82596 checks each passing frame for an address match. The 82596 will recognize its own unique address, one or more multicast addresses, or the broadcast address. If a match is found the 82596 stores the destination and source addresses and the length field in the next available RFD. It then begins filling the next available Data Buffer on the FBL, which is pointed to by the current RFD, with the data portion of the incoming frame. As one Data Buffer is filled, the 82596 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers to fit frames much shorter than the maximum allowable frame length. If AL-LOC = 1, or if the flexible memory structure is used, the addresses and length field can be placed in the Receive Buffer.

Once the entire frame is received without error, the 82596 does the following housekeeping tasks.

- The actual count field of the last Buffer Descriptor used to hold the frame just received is updated with the number of bytes stored in the associated Data Buffer.
- The next available Receive Frame Descriptor is fetched.
- The address of the next available Buffer Descriptor is written to the next available Receive Frame Descriptor.
- A frame received interrupt status bit is posted in the SCB.
- An interrupt is sent to the CPU.

If a frame error occurs, for example a CRC error, the 82596 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad frame. The 82596 will continue to receive frames without CPU help as long as Receive Frame Descriptors and Data Buffers are available.

82596 NETWORK MANAGEMENT AND DIAGNOSTICS

The behavior of data communication networks is normally very complex because of their distributed and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82596 has extensive diagnostic and network management functions that help improve reliability and testability. The 82596 reports on the following events after each frame is transmitted.

- Transmission successful.
- Transmission unsuccessful. Lost Carrier Sense.
- Transmission unsuccessful. Lost Clear to Send.
- Transmission unsuccessful. A DMA underrun occurred because the system bus did not keep up with the transmission.
- Transmission unsuccessful. The number of collisions exceeded the maximum allowed.
- Number of Collisions. The number of collisions experienced during the frame.
- Heartbeat Indicator. This indicates the presence of a heartbeat during the last Interframe Spacing (IFS) after transmission.

When configured to Save Bad Frames the 82596 checks each incoming frame and reports the following errors.

- CRC error. Incorrect CRC in a properly aligned frame.
- Alignment error. Incorrect CRC in a misaligned frame.
- Frame too short. The frame is shorter than the value configured for minimum frame length.
- Overrun. Part of the frame was not placed in memory because the system bus did not keep up with incoming data.
- Out of buffer. Part of the frame was discarded because of insufficient memory storage space.
- Receive collision. A collision was detected during reception.
- Length error. A frame not matching the frame length parameter was detected.

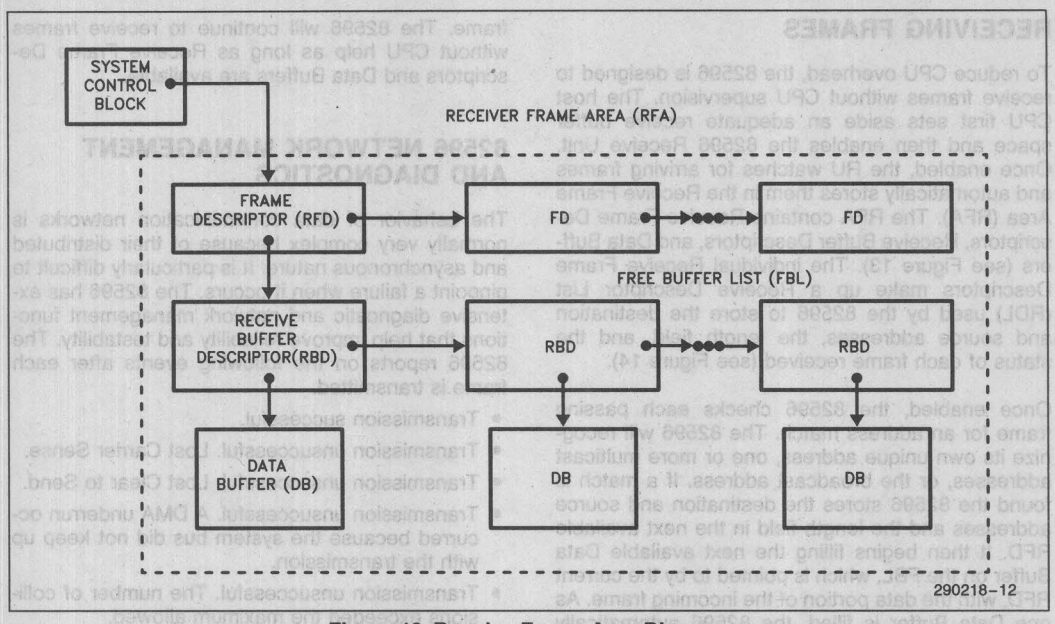


Figure 13. Receive Frame Area Diagram

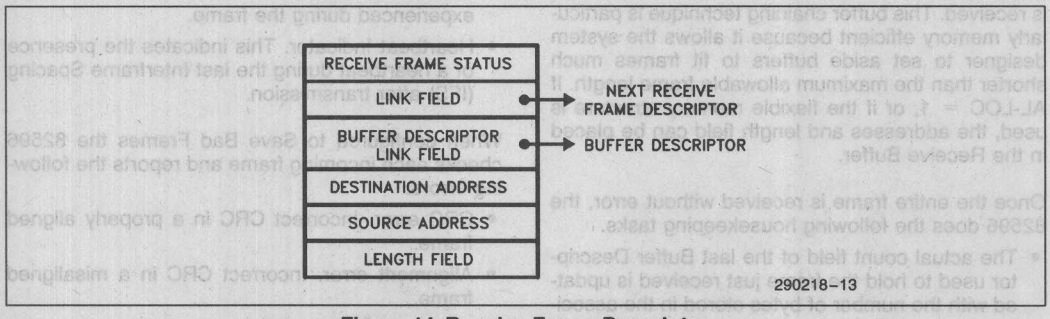


Figure 14. Receive Frame Descriptor

NETWORK PLANNING AND MAINTENANCE

To properly plan, operate, and maintain a communication network, the network management entity must accumulate information on network behavior. The 82596 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Information on network activity is provided in the status of each frame transmitted. The 82596 reports the following activity indicators after each frame.

- Number of collisions. The number of collisions the 82596 experienced while attempting to transmit the frame.
- Deferred transmission. During the first transmission attempt the 82596 had to defer to traffic on the link.

The 82596 updates its 32-bit statistical counters after each received frame that both passes address filtering and is longer than the Minimum Frame Length configuration parameter. The 82596 reports the following statistics.

- CRC errors. The number of well-aligned frames that experienced a CRC error.
- Alignment errors. The number of misaligned frames that experienced a CRC error.
- No resources. The number of frames that were discarded because of insufficient resources for reception.
- Overrun errors. The number of frames that were not completely stored in memory because the system bus did not keep up with incoming data.
- Receive Collision counter. The number of collisions detected during receive.
- Short Frame counter. The number of frames that were discarded because they were shorter than the configured minimum frame length.

The 82596 can be configured to Promiscuous mode. In this mode it captures all frames transmitted on the network without checking the Destination Address. This is useful when implementing a monitoring station to capture all frames for analysis.

A useful method of capturing frame headers is to use the Simplified memory mode, configure the 82596 to Save Bad Frames, and configure the 82596 to Promiscuous mode with space in the RFD allocated for specific number of receive data bytes. The 82596 will receive all frames and put them in the RFD. Frames that exceed the available space in the RFD will be truncated, the status will be updated, and the 82596 will retrieve the next RFD. This allows the user to capture the initial data bytes of each frame (for instance, the header) and discard the remainder of the frame.

The 82596 also has a monitor mode for network analysis. During normal operation the receive function enables the 82596 to receive frames that pass address filtering. These frames must have the Start of Frame Delimiter (SFD) field and must be longer than the absolute minimum frame length of 5 bytes (6 bytes in case of Multicast address filtering). Contents and status of the received frames are transferred to memory. The monitor function enables the 82596 to simply evaluate the incoming frames. The 82596 can monitor the frames that pass or do not pass the address filtering. It can also monitor frames which do not have the SFD fields. The 82596 can be configured to only keep statistical information about monitor frames. Three options are available in the Monitor mode. These options are selected by the two monitor mode configuration bits available in the configuration command.

When the first option is selected, the 82596 receives good frames that pass address filtering and transfers them to memory while monitoring frames that do not pass address filtering or are shorter than the minimum frame size (these frames are not transferred to memory). When this option is used the 82596 updates six counters: CRC errors, alignment errors, no resource errors, overrun errors, short frames and total good frames received.

When the second option is selected, the receive function is completely disabled. The 82596 monitors only those frames that pass address filterings and meet the minimum frame length requirement. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.

When the third option is selected, the receive function is completely disabled. The 82596 monitors all frames, including frames that do not have a Start Frame Delimiter. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected and total good frames.

82596 SOFTWARE COMPATIBILITY

The 82596 has a software-compatible state in which all its memory structures are compatible with the 82586 memory structure. This includes all the Action Commands, the Receive Frame Area (including the RFD, Buffer Descriptors, and Data Buffers), the System Control Block, and the initialization procedures. There are two minor differences between the 82596 in the 82586-Compatible memory structure and the 82586.

- When the internal and external loopback bits in the Configure command are set to 11 the 82596 is in external loopback and the $\overline{LPBK}$ pin is activated; in the 82586 this situation would produce internal loopback.
- During a Dump command both the 82596 and 82586 dump the same number of bytes; however, the data format is different.

address bus. Pins D_3–D_0 must be 0010; i.e., any alternative address must be aligned to 16-byte boundaries. All addresses sent to the 82596 must be word aligned, which means that all pointers and memory structures must start on an even address (A_0 = zero).

SYSTEM CONFIGURATION POINTER (SCP)

The SCP contains the sysbus byte and the location of the next structure of the initialization process, the ISCP. The following parameters are selected in the SYSBUS.

- The 82596 operation mode.
- The Bus Throttle timer triggering method.
- Lock enabled.
- Interrupt polarity.

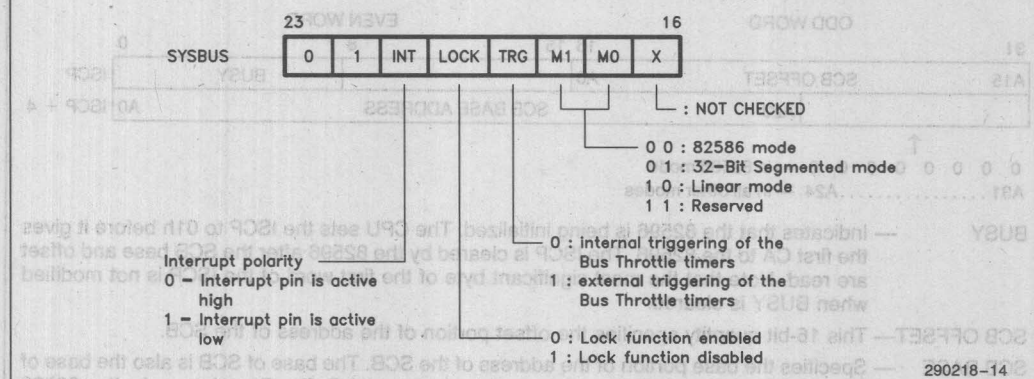
Byte ordering is determined by the LE/$\overline{BE}$ pin. LE/$\overline{BE}$ = 1 selects Little Endian byte ordering and LE/$\overline{BE}$ = 0 selects Big Endian byte ordering.

NOTE:
In the following, X indicates a bit not checked 82586 mode. This bit must be set to 0 in all other modes.

The following diagram illustrates the format of the SCP.

31 ODD WORD								16	15 EVEN WORD								0	
X	X	X	X	X	X	X	X	SYSBUS	0	0	0	0	0	0	0	0	0FFFFF4h
X	X	X	X	X	X	X	X	X X X X X X X X	X	X	X	X	X	X	X	X	X X X X X X X X	0FFFFF8h
A31 ..... A24								A23	ISCP ADDRESS								A0	0FFFFFCh

A31 ..... A24 are not checked in 82586 mode
X ..... X areas are not checked in 82586 mode; they must be 0 in all other modes.



ISCP ADDRESS— The physical address of the ISCP. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 15. The System Configuration Pointer

Writing the Sysbus

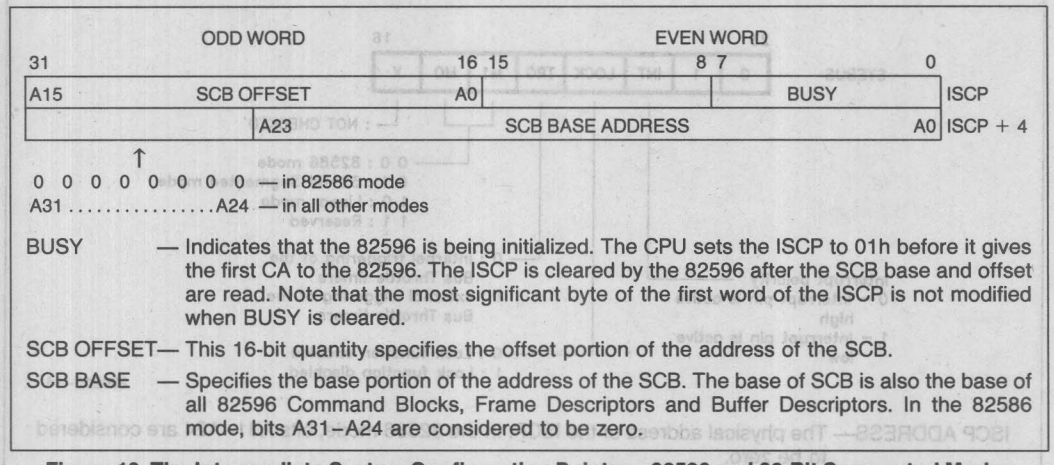
When writing the sysbus byte it is important to pay attention to the byte order.

- When a Little Endian processor is used, the sysbus byte is located at byte address 00FFFFF6h (or address $n+2$ if an alternative SCP address n was programmed).
- When a processor using Big Endian byte ordering is used, the sysbus, alternative SCP, and ISCP addresses will be different.
  - The sysbus byte is located at 00FFFFF5h.
  - If an alternative SCP address is programmed, the sysbus byte should be at byte address $n+1$.

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)

The ISCP indicates the location of the System Control Block. Often the SCP is in ROM and the ISCP is in RAM. The CPU loads the SCB address (or an equivalent data structure) into the ISCP and asserts CA. This Channel Attention signal causes the 82596 to begin its initialization procedure and to get the SCB address from the ISCP and SCP. In 82586 and 32-bit Segmented modes the SCP base address is also the base address of all Command Blocks, Frame Descriptors, and Buffer Descriptors (but not buffers). All these data structures must reside in one 64-KB segment; however, in Linear mode no such limitation is imposed.

The following diagram illustrates the ISCP format.

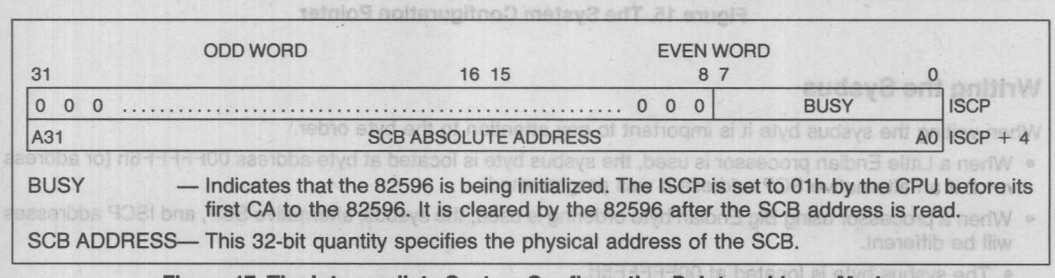


BUSY — Indicates that the 82596 is being initialized. The CPU sets the ISCP to 01h before it gives the first CA to the 82596. The ISCP is cleared by the 82596 after the SCB base and offset are read. Note that the most significant byte of the first word of the ISCP is not modified when BUSY is cleared.

SCB OFFSET— This 16-bit quantity specifies the offset portion of the address of the SCB.

SCB BASE — Specifies the base portion of the address of the SCB. The base of SCB is also the base of all 82596 Command Blocks, Frame Descriptors and Buffer Descriptors. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 16. The Intermediate System Configuration Pointer—82586 and 32-Bit Segmented Modes



BUSY — Indicates that the 82596 is being initialized. The ISCP is set to 01h by the CPU before its first CA to the 82596. It is cleared by the 82596 after the SCB address is read.

SCB ADDRESS— This 32-bit quantity specifies the physical address of the SCB.

Figure 17. The Intermediate System Configuration Pointer—Linear Mode.

INITIALIZATION PROCESS

The CPU sets up the SCP, ISCP, and the SCB structures, and, if desired, an alternative SCP address. It also sets BUSY to 01h. The 82596 is initialized when a Channel Attention signal follows a Reset signal, causing the 82596 to access the System Configuration Pointer. The sysbus byte, the operational mode, the bus throttle timer triggering method, the interrupt polarity, and the state of $\overline{LOCK}$ are read. After reset the Bus Throttle timers are essentially disabled—the T-ON value is infinite, the T-OFF value is zero. After the SCP is read, the 82596 reads the ISCP and saves the SCB address. In 82586 and 32-bit Segmented modes this address is represented as a base address plus the offset (this base address is also the base address of all the control blocks). In Linear mode the base address is also an absolute address. The 82596 clears BUSY, sets CX and CNR to equal 1 in the SCB, clears the SCB command word, sends an interrupt to the CPU, and awaits another Channel Attention signal. RESET configures the 82596 to its default state before CA is asserted.

CONTROLLING THE 82596CA

The host CPU controls the 82596 with the commands, data structures, and methods described in this section. The CPU and the 82596 communicate through shared memory structures. The 82596 contains two independent units: the Command Unit and the Receive Unit. The Command Unit executes commands from the CPU, and the Receive Unit handles frame reception. These two units are controlled and monitored by the CPU through a shared memory structure called the System Control Block (SCB). The CPU and the 82596 use the CA and INT signals to communicate with the SCB.

82596 CPU ACCESS INTERFACE ($\overline{PORT}$)

The 82596 has a CPU access interface that allows the host CPU to do four things.

- Write an alternative System Configuration Pointer address.
- Write an alternative Dump area pointer and perform Dump.
- Execute a software reset.
- Execute a self-test.

The following events initiate the CPU access state.

- Presence of an address on the $D_{31}-D_4$ data bus pins.
- The D_3-D_0 pins are used to select one of the four functions.
- The $\overline{PORT}$ input pin is asserted, as in a regular write cycle.

NOTE.
The SCP Dump and Self-Test addresses must be 16-byte aligned.

The 82596 requires two 16-bit write cycles for a port command. The first write holds the internal machines and reads the first 16 bits; the second activates the $\overline{PORT}$ command and reads the second 16 bits.

The $\overline{PORT}$ Reset is useful when only the 82596 needs to be reset. The CPU must wait for 10-system and 5-serial clocks before issuing another CA to the 82596; this new CA begins a new initialization process.

The Dump function is useful for troubleshooting No Response problems. If the chip is in a No Response state, the $\overline{PORT}$ Dump operation can be executed and a $\overline{PORT}$ Reset can be used to reinitialize the 82596 without disturbing the rest of the system.

The Self-Test function can be used for board testing; the 82596 will execute a self-test and write the results to memory.

Table 2. $\overline{PORT}$ Function Selection

Function	D31 ..... D4 Addresses and Results			D3	D2	D1	D0
Reset	A31	Don't Care	A4	0	0	0	0
Self-Test	A31	Self-Test Results Address	A4	0	0	0	1
SCP	A31	Alternative SCP Address	A4	0	0	1	0
Dump	A31	Dump Area Pointer	A4	0	0	1	1

must wholly reside within one segment. If the 82596 encounters a memory structure
restriction, the 82596 will fetch the next contiguous location in memory (beyond the segment).

- Linear Mode
  - A Linear address is a single 32-bit entity.
  - There are no Segmented addresses.

Linear addresses are primarily used to address transmit and receive data buffers. In the 82586 and 32-bit Segmented modes, segmented addresses (base plus offset) are used for all Command Blocks, Buffer Descriptors, Frame Descriptors, and System Control Blocks. When using Segmented addresses, only the offset portion of the entity being addressed is specified in the block. The base for all offsets is the same—that of the SCB. See Table 1.

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING

The 82596 supports both Little Endian and Big Endian byte ordering for its memory structures.

The 82596 supports Big Endian byte ordering for word and byte entities. Dword entities are not supported with Big Endian byte ordering. This results in slightly different 82596 memory structures for Big Endian operation. These structures are defined in the *82596 User's Manual.*

NOTE

All 82596 memory entities must be word or dword aligned.

An example of a dword entity is a frame descriptor command/status dword, whereas the raw data of the frame are byte entities. Both 32- and 16-bit buses are supported. When a 16-bit bus is used with Big Endian memory organization, data lines D_{15}–D_0 are used. The 82596 has an internal crossover that handles these swap operations.

COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block is associated with each Action Command. The CU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- **Suspended.** The CU is not executing a command; however, it is associated with a CB on the list.
- **Active.** The CU is executing an Action Command and pointing to its CB.

The CPU can affect CU operation in two ways: by issuing a CU Control Command or by setting bits in the Command word of the Action Command.

RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory. The RU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The RU has no memory resources and is discarding incoming frames. This is the initial state.
- **No Resources.** The RU has no memory resources and is discarding incoming frames. This state differs from Idle in that the RU accumulates statistics on the number of discarded frames.
- **Suspended.** The RU has memory available for storing frames, but is discarding them. The suspend state can only be reached if the CPU forces this through the SCB or sets the suspend bit in the RFD.
- **Ready.** The RU has memory available and is storing incoming frames.

The CPU can affect RU operation in three ways: by issuing an RU Control Command, by setting bits in the Frame Descriptor Command word of the frame being received, or by setting the EL bit of the current buffer's Buffer Descriptor.

SYSTEM CONTROL BLOCK (SCB)

The SCB is a memory block that plays a major role in communications between the CPU and the 82596. Such communications include the following.

- Commands issued by the CPU
- Status reported by the 82596

Control commands are sent to the 82596 by writing them into the SCB and then asserting CA. The 82596 examines the command, performs the required action, and then clears the SCB command word. Control commands perform the following types of tasks.

- Operation of the Command Unit (CU). The SCB controls the CU by specifying the address of the Command Block List (CBL) and by starting, suspending, resuming, or aborting execution of CBL commands.
- Operation of the Bus Throttle. The SCB controls the Bus Throttle timers by providing them with new values and sending the Load and Start timer commands. The timers can be operated in both the 32-bit Segmented and Linear modes.
- Reception of frames by the Receive Unit (RU). The SCB controls the RU by specifying the address of the Receive Frame Area and by starting, suspending, resuming, or aborting frame reception.
- Acknowledgment of events that cause interrupts.
- Resetting the chip.

The 82596 sends status reports to the CPU via the System Control Block. The SCB contains four types of status reports.

- The cause of the current interrupts. These interrupts are caused by one or more of the following 82596 events.
  - The Command Unit completes an Action Command that has its I bit set.
  - The Receive Unit receives a frame.
  - The Command Unit becomes inactive.
  - The Receive Unit becomes not ready.
- The status of the Command Unit.
- The status of the Receive Unit.
- Status reports from the 82596 regarding reception of corrupted frames.

Events can be cleared only by CPU acknowledgment. If some events are not acknowledged by the ACK field the Interrupt signal (INT) will be reissued after Channel Attention (CA) is processed. Furthermore, if a new event occurs while an interrupt is set, the interrupt is temporarily cleared to trigger edge-triggered interrupt controllers.

The CPU uses the Channel Attention line to cause the 82596 to examine the SCB. This signal is trailing-edge triggered—the 82596 latches CA on the trailing edge. The latch is cleared by the 82596 before the SCB control command is read.

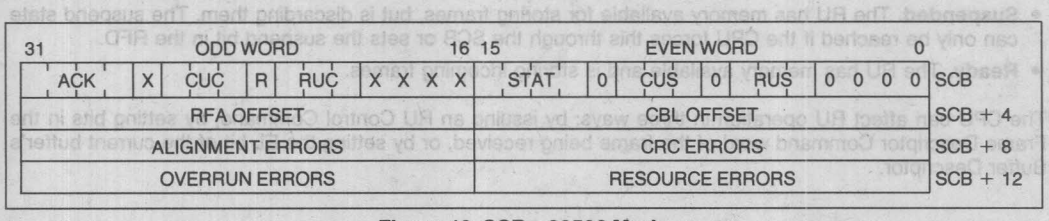


Figure 18. SCB—82586 Mode

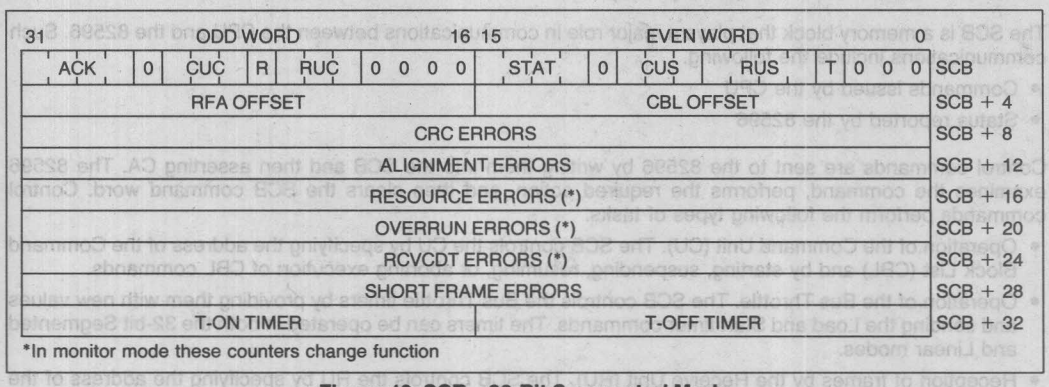


Figure 19. SCB—32-Bit Segmented Mode

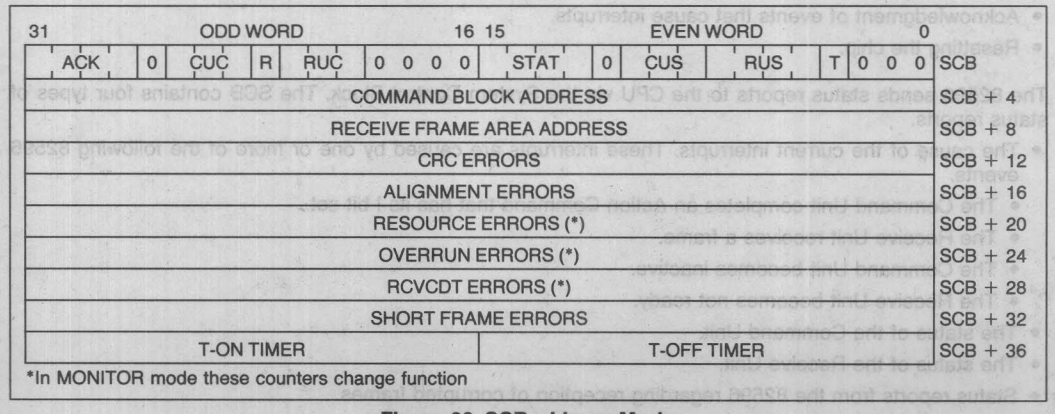


Figure 20. SCB—Linear Mode

Command Word

31									16	
ACK	0	CUC	R	RUC	0	0	0	0		SCB + 2

These bits specify the action to be performed as a result of a CA. This word is set by the CPU and cleared by the 82596. Defined bits are:

Bit 31 ACK-CX — Acknowledges that the CU completed an Action Command.

Bit 30 ACK-FR — Acknowledges that the RU received a frame.

Bit 29 ACK-CNA — Acknowledges that the Command Unit became not active.

Bit 28 ACK-RNR — Acknowledges that the Receive Unit became not ready.

Bits 24–26 CUC — (3 bits) This field contains the command to the Command Unit. Valid values are:

- 0 — NOP (does not affect current state of the unit).
- 1 — Start execution of the first command on the CBL. If a command is executing, complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET (address).
- 2 — Resume the operation of the Command Unit by executing the next command. This operation assumes that the Command Unit has been previously suspended.
- 3 — Suspend execution of commands on CBL after current command is complete.
- 4 — Abort current command immediately.
- 5 — Loads the Bus Throttle timers so they will be initialized with their new values after the active timer (T-ON or T-OFF) reaches Terminal Count. If no timer is active new values will be loaded immediately. This command is not valid in 82586 mode.
- 6 — Loads and immediately restarts the Bus Throttle timers with their new values. This command is not valid in 82586 mode.
- 7 — Reserved.

Bit 23 RESET — Reset chip (logically the same as hardware RESET).

Bits 20–22 RUC — (3 bits) This field contains the command to the Receive Unit. Valid values are:

- 0 — NOP (does not alter current state of unit).
- 1 — Start reception of frames. The beginning of the RFA is contained in the RFA OFFSET (address). If a frame is being received complete reception before starting.
- 2 — Resume frame reception (only when in suspended state).
- 3 — Suspend frame reception. If a frame is being received complete its reception before suspending.
- 4 — Abort receiver operation immediately.
- 5–7 — Reserved.

Bit 12 RNR — The Receive Unit left the Ready state.

Bits 8–10 CUS — (3 bits) This field contains the status of the command unit. Valid values are:

0 — Idle
1 — Suspended
2 — Active
3–7 — Not used

Bits 4–7 RUS — This field contains the status of the receive unit. Valid values are:

0h (0000) — Idle
1h (0001) — Suspended
2h (0010) — No Resources. This bit indicates both no resources due to lack of RFDs in the RDL and no resources due to lack of RBDs in the FBL.
4h (0100) — Ready
8h (1000) — No more RBDs (not in the 82586 mode)
Ah (1010) — No resources due to no more RBDs (not in the 82586 mode).
No other combinations are allowed

Bit 3 T — Bus Throttle timers loaded (not in 82586 mode).

SCB OFFSET ADDRESSES

CBL Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the first Command Block on the CBL. In Linear mode it is a 32-bit linear address for the first Command Block on the CBL. It is accessed only if CUC equals Start.

RFA Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the Receive Frame Area. In Linear mode it is a 32-bit linear address for the Receive Frame Area. It is accessed only if RUC equals Start.

SCB STATISTICAL COUNTERS

Statistical Counter Operation

- The CPU is responsible for clearing all error counters before initializing the 82596. The 82596 updates these counters by reading them, adding 1, and then writing them back to the SCB.
- The counters are wraparound counters. After reaching FFFFFFFFh the counters wrap around to zero.
- The 82596 updates the required counters for each frame. It is possible for more than one counter to be updated; multiple errors will result in all affected counters being updated.
- The 82596 executes the read-counter/increment/write-counter operation without relinquishing the bus (locked operation). This is to ensure that no logical contention exists between the 82596 and the CPU due to both attempting to write to the counters simultaneously. In the dual-port memory configuration the CPU should not execute any write operation to a counter if $\overline{\text{LOCK}}$ is asserted.
- The counters are 32-bits wide and their behavior is fully compatible with the IEEE 802.3 standard. The 82596 supports all relevant statistics (mandatory, optional, and desired) through the status of the transmit and receive header and directly through SCB statistics.

CRCERRS

This 32-bit quantity contains the number of aligned frames discarded because of a CRC error. This counter is updated, if needed, regardless of the RU state.

ALNERRS

This 32-bit quantity contains the number of frames that both are misaligned (i.e., where $\overline{\text{CRS}}$ deasserts on a nonoctet boundary) and contain a CRC error. The counter is updated, if needed, regardless of the RU state.

SHRTFRM

This 32-bit quantity contains the number of received frames shorter than the minimum frame length.

The last three counters change function in monitor mode.

RSCERRS

This 32-bit quantity contains the number of good frames discarded because there were no resources to contain them. Frames intended for a host whose RU is in the No Receive Resources state, fall into this category. This counter is updated only if the RU is in the No Resources state. When in Monitor mode this counter counts the total number of frames—good and bad.

OVRNERRS

This 32-bit quantity contains the number of frames known to be lost because the local system bus was not available. If the traffic problem lasts longer than the duration of one frame, the frames that follow the first are lost without an indicator, and they are not counted. This counter is updated, if needed, regardless of the RU state.

RCVCDT

This 32-bit quantity contains the number of collisions detected during frame reception. In Monitor mode this counter counts the total number of good frames.

ACTION COMMANDS AND OPERATING MODES

This section lists all the Action Commands of the Command Unit Command Block List (CBL). Each command contains the Command field, the Status and Control fields, the link to the next Action Command, and any command-specific parameters. There are three basic types of action commands: 82596 Configuration and Setup, Transmission, and Diagnostics. The following is a list of the actual commands.

- NOP
- Individual Address Setup
- Configure
- MC Setup
- Transmit
- TDR
- Dump
- Diagnose

The 82596 has three addressing modes. In the 82586 mode all the Action Commands look exactly like those of the 82586.

- **82586 Mode.** The 82596 software and memory structure is compatible with the 82586.
- **32-Bit Segmented Mode.** The 82596 can access the entire system memory and use the two new memory structures—Simplified and Flexible—while still using the segmented approach. This does not require any significant changes to existing software.
- **Linear Mode.** The 82596 operates in a flat, linear, 4 gigabyte memory space without segmentation. It can also use the two new memory structures.

In the 32-bit Segmented mode there are some differences between the 82596 and 82586 action commands, mainly in programming and activating new 82596 features. Those bits marked "don't care" in the compatible mode are not checked; however, we strongly recommend that those bits all be zeroes; this will allow future enchancements and extensions.

In the Linear mode all of the address offsets become 32-bit address pointers. All new 82596 features are accessible in this mode, and all bits previously marked "don't care" must be zeroes.

The Action Commands, and all other 82596 memory structures, must begin on even byte boundaries, i.e., they must be word aligned.

NOP

This command results in no action by the 82596 except for those performed in the normal command processing. It is used to manipulate the CBL manipulation. The format of the NOP command is shown in Figure 21.

NOP—82586 and 32-Bit Segmented Modes

31																16	15															0	
ODD WORD																	EVEN WORD																
EL	S	I	X	X	X	X	X	X	X	X	X	X	0	0	0	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	A15																A0	4
LINK OFFSET																																	

NOP—Linear Mode

31																16	15															0	
ODD WORD																	EVEN WORD																
EL	S	I	0	0	0	0	0	0	0	0	0	0	0	0	0	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A31																																A0	4
LINK ADDRESS																																	

Figure 21

where:

LINK POINTER — In the 82586 or 32-bit Segmented modes this is a 16-bit offset to the next Command Block. In the Linear mode this is the 32-bit address of the next Command Block.

EL — If set, this bit indicates that this command block is the last on the CBL.

S — If set to one, suspend the CU upon completion of this CB.

I — If set to one, the 82596 will generate an interrupt after execution of the command is complete. If I is not set to one, the CX bit will not be set.

CMD (bits 16–18) — The NOP command. Value: 0h.

Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).

C — This bit indicates the execution status of the command. The CPU initially resets it to zero when the Command Block is placed on the CBL. Following a command Completion, the 82596 will set it to one.

B — This bit indicates that the 82596 is currently executing the NOP command. It is initially reset to zero by the CPU. The 82596 sets it to one when execution begins and to zero when execution is completed. This bit is also set when the 82596 prefetches the command.

NOTE:
The C and B bits are modified in one operation.

OK — Indicates that the command was executed without error. If set to one no error occurred (command executed OK). If zero an error occured.

Individual Address Setup

This command is used to load the 82596 with the Individual Address. This address is used by the 82596 for inserting the Source Address during transmission and recognizing the Destination Address during reception. After RESET, and prior to Individual Address Setup Command execution, the 82596 assumes the Broadcast Address is the Individual Address in all aspects, i.e.:

- This will be the Individual Address Match reference.
- This will be the Source Address of a transmitted frame (for AL-LOC=0 mode only).

The format of the Individual Address Setup command is shown in Figure 22.

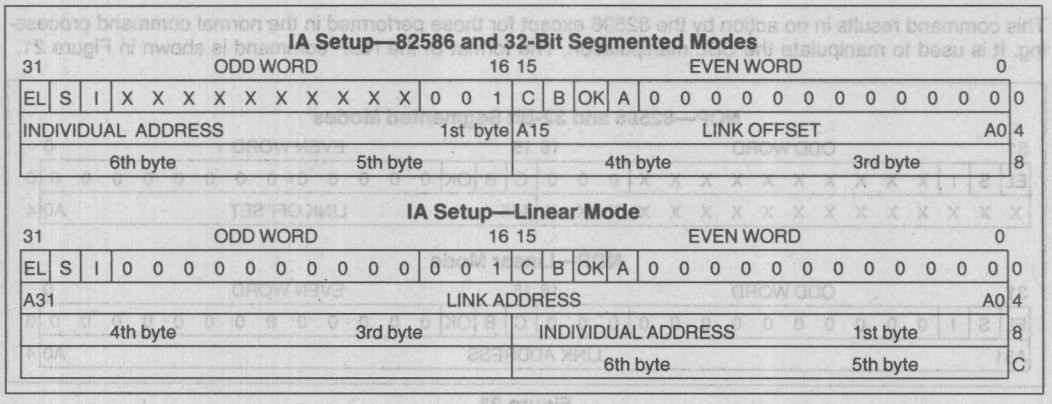


Figure 22

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).

CMD (bits 16–18) — The Address Setup command. Value: 1h.

INDIVIDUAL ADDRESS — The individual address of the node, 0 to 6 bytes long.

The least significant bit of the Individual Address must be zero for Ethernet (see the Command Structure). However, no enforcement of 0 is provided by the 82596. Thus, an Individual Address with 1 as its least significant bit is a valid Individual Address in all aspects.

The default address length is 6 bytes long, as in 802.3. If a different length is used the IA Setup command should be executed after the Configure command.

Configure

The Configure command loads the 82596 with its operating parameters. It allows changing some of the parameters by specifying a byte count less than the maximum number of configuration bytes (12 in the 82586 mode, 16 in the 32-Bit Segmented and Linear modes). The 82596 configuration depends on its mode of operation.

- In the 82586 mode the maximum number of configuration bytes is 12. Any number larger than 12 will be reduced to 12 and any number less than 4 will be increased to 4.
- The additional features of the serial side are disabled in the 82586 mode.
- In both the 32-Bit Segmented and Linear modes there are four additional configuration bytes, which hold parameters for additional 82596 features. If these parameters are not accessed, the 82596 will follow their default values.
- For more detailed information refer to the 32-Bit LAN Components User's Manual.

The format of the Configure command is shown in Figure 23, 24 and 25.

31 ODD WORD 16	15 EVEN WORD 0	
EL S I X X X X X X X X X X 0 1 0	C B OK A 0 0 0 0 0 0 0 0 0 0 0 0	0
Byte 1 / Byte 0	A15 LINK OFFSET A0	4
Byte 5 / Byte 4	Byte 3 / Byte 2	8
Byte 9 / Byte 8	Byte 7 / Byte 6	12
X X X X X X X X X X X X X X X X	X X X X X X X X / Byte 10	16

Figure 23. CONFIGURE—82586 Mode

31 ODD WORD 16	15 EVEN WORD 0	
EL S I 0 0 0 0 0 0 0 0 0 0 0 1 0	C B OK A 0 0 0 0 0 0 0 0 0 0 0 0	0
Byte 1 / Byte 0	A15 LINK OFFSET A0	4
Byte 5 / Byte 4	Byte 3 / Byte 2	8
Byte 9 / Byte 8	Byte 7 / Byte 6	12
Byte 13 / Byte 12	Byte 11 / Byte 10	16

Figure 24. CONFIGURE—32-Bit Segmented Mode

31 ODD WORD 16	15 EVEN WORD 0	
EL S I 0 0 0 0 0 0 0 0 0 0 0 1 0	C B OK A 0 0 0 0 0 0 0 0 0 0 0 0	0
A31 LINK ADDRESS A0		4
Byte 3 / Byte 2	Byte 1 / Byte 0	8
Byte 7 / Byte 6	Byte 5 / Byte 4	12
Byte 11 / Byte 10	Byte 9 / Byte 8	16
X X X X X X X X X X X X X X X X	Byte 13 / Byte 12	20

Figure 25. CONFIGURE—Linear Mode

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If 1, then the command was aborted and if necessary it should be repeated. If this bit is 0, the command was not aborted.

Bits 19–28 — Reserved (zero in the 32-Bit Segmented and Linear Modes)

CMD (bits 16–18) — The CONFIGURE command. Value: 2h.

The interpretation of the fields follows:

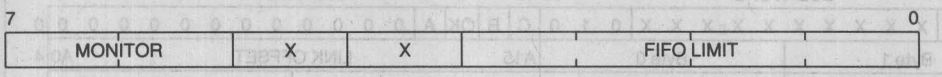
7	6	5	4	3	2	1	0
P	X	X	X	BYTE COUNT			

BYTE 0

BYTE CNT (Bits 0–3) — Byte Count. Number of bytes, including this one, that hold parameters to be configured.

PREFETCHED (Bit 7) — Enable the 82596 to write the prefetched bit in all prefetch RBDs.

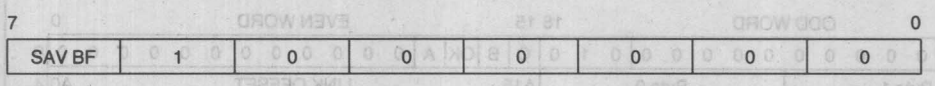
NOTE:
The P bit is valid only in the new memory structure modes. In 82586 mode this bit is disabled (i.e., no prefetched mark).



BYTE 1

FIFO Limit (Bits 0–3)	FIFO limit.
MONITOR # (Bits 6–7)	Receive monitor options. If the Byte Count of the configure command is less than 12 bytes then these Monitor bits are ignored.

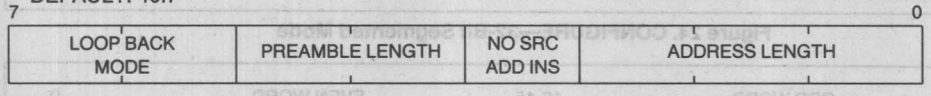
DEFAULT: C8h



BYTE 2

SAV BF (Bit 7)	0—Received bad frames are not saved in the memory. 1—Received bad frames are saved in the memory.

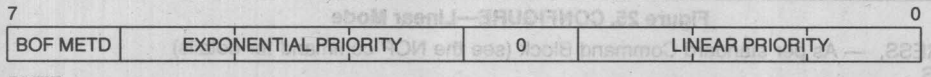
DEFAULT: 40h



BYTE 3

ADR LEN (Bits 0–2)	Address length (any kind).
NO SCR ADD INS (Bit 3)	No Source Address Insertion. In the 82586 this bit is called AL LOC.
PREAM LEN (Bits 4–5)	Preamble length.
LP BCK MODE (Bits 6–7)	Loopback mode.

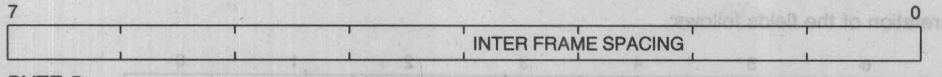
DEFAULT: 26h



BYTE 4

LIN PRIO (Bits 0–2)	Linear Priority.
EXP PRIO (Bits 4–6)	Exponential Priority.
BOF METD (Bit 7)	Exponential Backoff method.

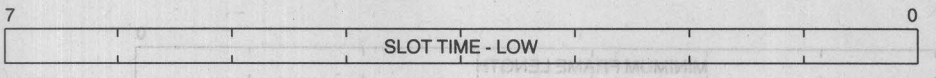
DEFAULT: 00h



BYTE 5

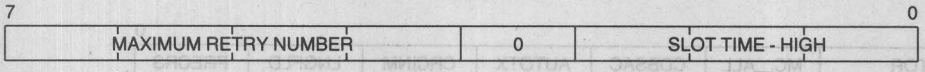
INTERFRAME SPACING	Interframe spacing.

DEFAULT: 60h



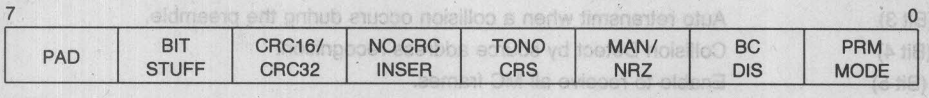
BYTE 6

SLOT TIME (L)	Slot time, low byte.
DEFAULT: 00h	



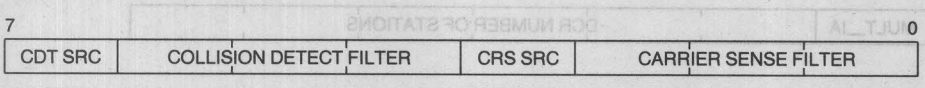
BYTE 7

SLOT TIME (H) (Bits 0–2)	Slot time, high part.
RETRY NUM (Bits 4–7)	Number of transmission retries on collision.
DEFAULT: F2h	



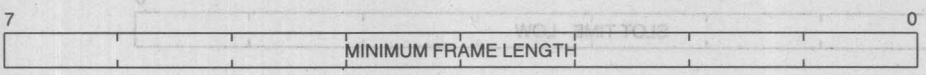
BYTE 8

PRM (Bit 0)	Promiscuous mode.
BC DIS (Bit 1)	Broadcast disable.
MANCH/NRZ (Bit 2)	Manchester or NRZ encoding. See specific timing requirements for TXC in Manchester mode.
TONO CRS (Bit 3)	Transmit on no CRS.
NOCRC INS (Bit 4)	No CRC insertion.
CRC-16/CRC-32 (Bit 5)	CRC type.
BIT STF (Bit 6)	Bit stuffing.
PAD (Bit 7)	Padding.
DEFAULT: 00h	



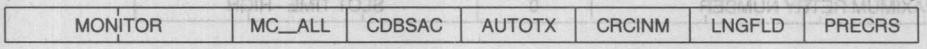
BYTE 9

CRSF (Bits 0–2)	Carrier Sense filter (length).
CRS SRC (Bit 3)	Carrier Sense source.
CDTF (Bits 4–6)	Collision Detect filter (length).
CDT SRC (Bit 7)	Collision Detect source.
DEFAULT: 00h	



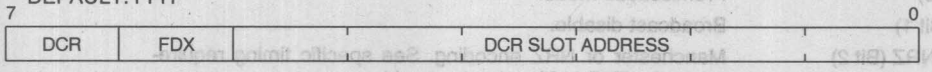
BYTE 10

MIN FRAME LEN	Minimum frame length.
DEFAULT: 40h	



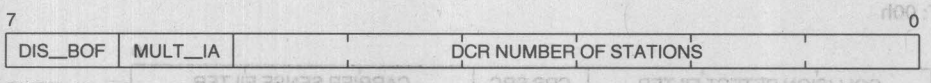
BYTE 11

PRECRS (Bit 0)	Preamble until Carrier Sense
LNGFLD (Bit 1)	Length field. Enables padding at the End-of-Carrier framing (802.3).
CRCINM (Bit 2)	Rx CRC appended to the frame in memory.
AUTOTX (Bit 3)	Auto retransmit when a collision occurs during the preamble.
CDBSAC (Bit 4)	Collision Detect by source address recognition.
MC_ALL (Bit 5)	Enable to receive all MC frames.
MONITOR (Bits 6–7)	Receive monitor options.
DEFAULT: FFH	



BYTE 12

DCR SLOT ADDRESS (Bits 0–5)	Station index in DCR mode.
FDX (Bit 6)	Enables Full Duplex operation.
DCR (Bit 7)	Enables Deterministic collision resolution.
DEFAULT: 00h	



BYTE 13

DCR NUMBER OF STATIONS (Bits 0–5)	Number of stations in DCR mode.
MULT_IA (Bit 6)	Multiple individual address.
DIS_BOF (Bit 7)	Disable the backoff algorithm.
DEFAULT: 3Fh	

*	CRC IN MEMORY	1	CRC Not Transferred to Memory
	CRC-16/CRC-32	**0	CRC-32
	CRS FILTER	0	0 Bit Times
	CRS SRC	0	External CRS
*	DCR	0	Disable DCR Protocol
*	DCR Slot Number	0	DCR Disabled
*	DCR Number of Stations	63	Stations
*	DISBOF	0	Backoff Enabled
	EXT LOOPBACK	0	Disabled
	EXPONENTIAL PRIORITY	**0	802.3 Algorithm
	EXPONENTIAL BACKOFF METHOD	**0	802.3 Algorithm
*	FULL DUPLEX (FDX)	0	CSMA/CD Protocol (No FDX)
	FIFO THRESHOLD	8	TX: 32 Bytes, RX: 64 Bytes
	INT LOOPBACK	0	Disabled
	INTERFRAME SPACING	**96	Bit Times
	LINEAR PRIORITY	**0	802.3 Algorithm
*	LENGTH FIELD	1	Padding Disabled
	MIN FRAME LENGTH	**64	Bytes
*	MC ALL	1	Disabled
*	MONITOR	11	Disabled
	MANCHESTER/NRZ	0	NRZ
*	MULTI IA	0	Disabled
	NUMBER OF RETRIES	**15	Maximum Number of Retries
	NO CRC INSERTION	0	CRC Appended to Frame
	PREFETCH BIT IN RBD	0	Disabled (Valid Only in New Modes)
	PREAMBLE LENGTH	**7	Bytes
*	Preamble Until CRS	1	Disabled
	PROMISCUOUS MODE	0	Address Filter On
	PADDING	0	No Padding
	SLOT TIME	**512	Bit Times
	SAVE BAD FRAME	0	Discards Bad Frames
	TRANSMIT ON NO CRS	0	Disabled

NOTES

1. This configuration setup is compatible with the IEEE 802.3 specification.
2. The Asterisk "*" signifies a new configuration parameter not available in the 82586.
3. The default value of the Auto retransmit configuration parameter is enabled(1).
4. Double Asterisk "**" signifies IEEE 802.3 requirements.

Multicast-Setup

This command is used to load the 82596 with the Multicast-IDs that should be accepted. As noted previously, the filtering done on the Multicast-IDs is not perfect and some unwanted frames may be accepted. This command resets the current filter and reloads it with the specified Multicast-IDs. The format of the Multicast-addresses setup command is:

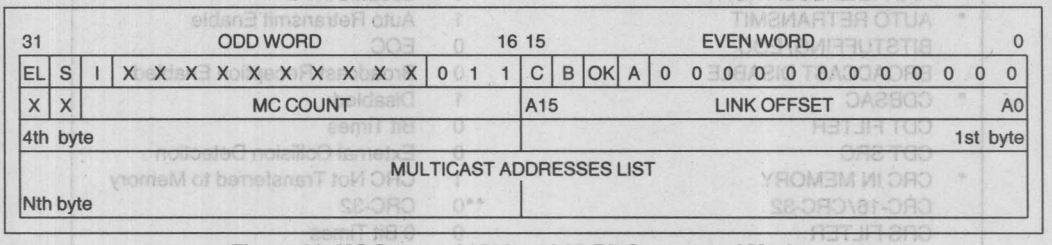


Figure 26. MC Setup—82586 and 32-Bit Segmented Modes

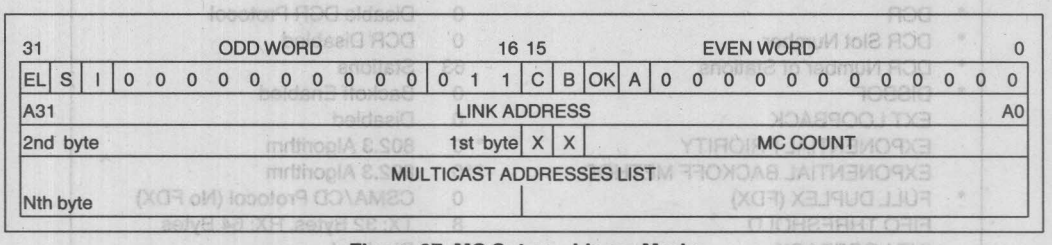


Figure 27. MC Setup—Linear Mode

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If one, then the command was aborted and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in both the 32-Bit Segmented and Linear Modes).

CMD (bits 16–18) — The MC SETUP command value: 3h.

MC-CNT — This 14-bit field indicates the number of bytes in the MC LIST field. The MC CNT must be a multiple of the ADDR LEN; otherwise, the 82596 reduces the MC CNT to the nearest ADDR LEN multiple. MC CNT = 0 implies resetting the Hash table which is equivalent to disabling the Multicast filtering mechanism.

MC LIST — A list of Multicast Addresses to be accepted by the 82596. The least significant bit of each MC address must be 1.

NOTE:

The list is sequential; i.e., the most significant byte of an address is immediately followed by the least significant byte of the next address.

— When the 82596 is configured to recognize multiple Individual Address (Multi-IA), the MC-Setup command is also used to set up the Hash table for the individual address.

The least significant bit in the first byte of each IA address must be 0.

Transmit

This command is used to transmit a frame of user data onto the serial link. The format of a Transmit command is as follows.

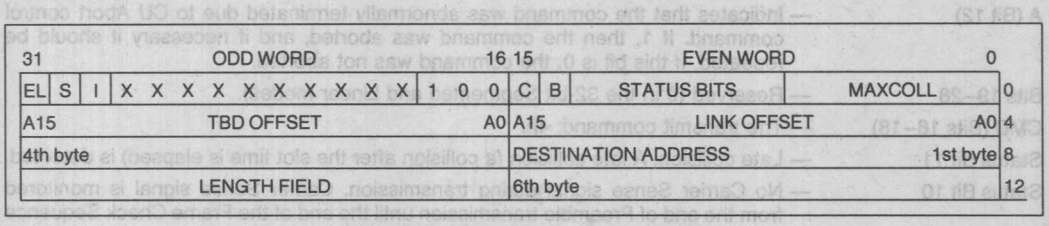


Figure 28. TRANSMIT—82586 Mode

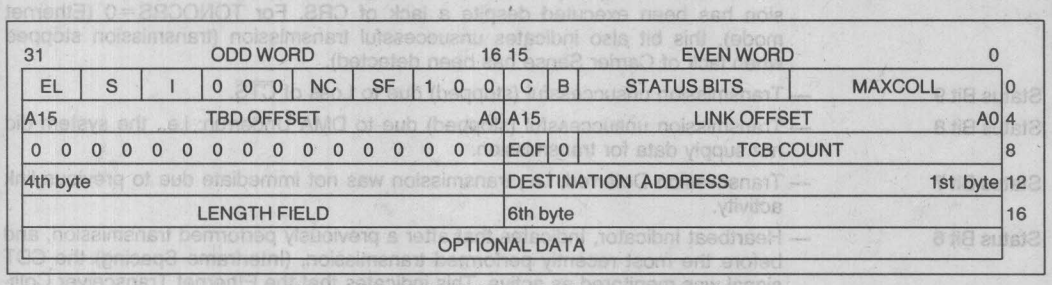


Figure 29. TRANSMIT—32-Bit Segmented Mode

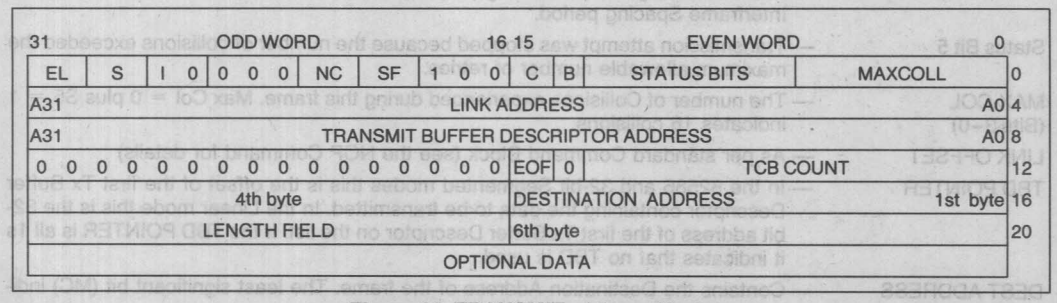
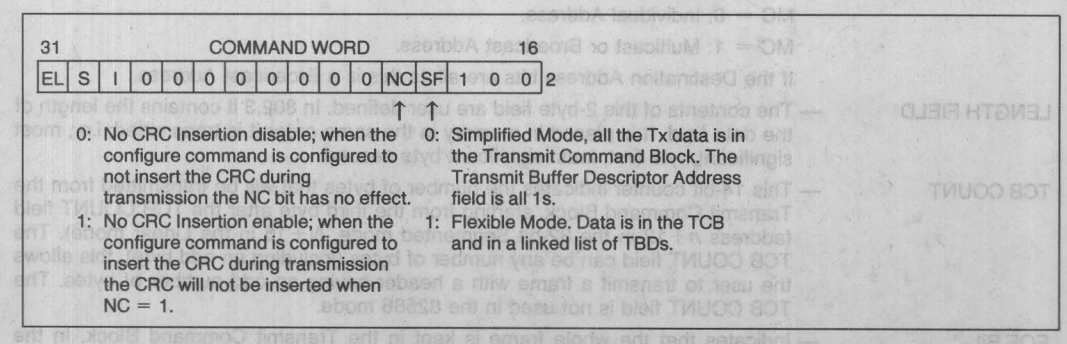


Figure 30. TRANSMIT—Linear Mode



where:

EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
OK (Bit 13)	— Error free completion.
A (Bit 12)	— Indicates that the command was abnormally terminated due to CU Abort control command. If 1, then the command was aborted, and if necessary it should be repeated. If this bit is 0, the command was not aborted.
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear modes).
CMD (Bits 16–18)	— The transmit command: 4h.
Status Bit 11	— Late collision. A late collision (a collision after the slot time is elapsed) is detected.
Status Bit 10	— No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense mode) it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).
Status Bit 9	— Transmission unsuccessful (stopped) due to Loss of $\overline{\text{CTS}}$.
Status Bit 8	— Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.
Status Bit 7	— Transmission Deferred, i.e., transmission was not immediate due to previous link activity.
Status Bit 6	— Heartbeat Indicator, Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing properly. The Heartbeat is monitored during the Interframe Spacing period.
Status Bit 5	— Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.
MAX-COL (Bits 3–0)	— The number of Collisions experienced during this frame. Max Col = 0 plus S5 = 1 indicates 16 collisions.
LINK OFFSET	— As per standard Command Block (see the NOP Command for details)
TBD POINTER	— In the 82586 and 32-bit Segmented modes this is the offset of the first Tx Buffer Descriptor containing the data to be transmitted. In the Linear mode this is the 32-bit address of the first Tx Buffer Descriptor on the list. If the TBD POINTER is all 1s it indicates that no TBD is used.
DEST ADDRESS	— Contains the Destination Address of the frame. The least significant bit (MC) indicates the address type. MC = 0: Individual Address. MC = 1: Multicast or Broadcast Address. If the Destination Address bits are all 1s this is a Broadcast Address.
LENGTH FIELD	— The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is transmitted; i.e., most significant byte first, least significant byte second.
TCB COUNT	— This 14-bit counter indicates the number of bytes that will be transmitted from the Transmit Command Block, starting from the third byte after the TCB COUNT field (address $n+12$ in the 32-bit Segmented mode, $N+16$ in the Linear mode). The TCB COUNT field can be any number of bytes (including an odd byte), this allows the user to transmit a frame with a header having an odd number of bytes. The TCB COUNT field is not used in the 82586 mode.
EOF Bit	— Indicates that the whole frame is kept in the Transmit Command Block. In the Simplified memory model it must be always asserted.

The interpretation of what is transmitted depends on the No Source Address insertion configuration bit and the memory model being used.

NOTES

1. The Destination Address and the Length Field are sequential. The Length Field immediately follows the most significant byte of the Destination Address.
2. In case the 82596 is configured with No Source Address insertion bit equal to 0, the 82596 inserts its configured Source Address in the transmitted frame.
   - In the 82586 mode, or when the Simplified memory model is used, the Destination and Length fields of the transmitted frame are taken from the Transmit Command Block.
   - If the FLEXIBLE memory model is used, the Destination and Length fields of the transmitted frame can be found either in the TCB or TBD, depending on the TCB COUNT.
3. If the 82596 is configured with the Address/Length Field Location equal to 1, the 82596 does not insert its configured Source Address in the transmitted frame. The first (2 × Address Length) + 2 bytes of the transmitted frame are interpreted as Destination Address, Source Address, and Length fields respectively. The location of the first transmitted byte depends on the operational mode of the 82596:
   - In the 82586 mode, it is always the first byte of the first Tx Buffer.
   - In both the 32-bit Segmented and Linear modes it depends on the SF bit and TCB COUNT:
     - — In the Simplified memory mode the first transmitted byte is always the third byte after the TCB COUNT field.
     - — In the Flexible mode, if the TCB COUNT is greater than 0 then it is the third byte after the TCB COUNT field. If TCB COUNT equals 0 then it is first byte of the first Tx Buffer.
   - Transmit frames shorter than six bytes are invalid. The transmission will be aborted (only in 82586 mode) because of a DMA Underrun.
4. Frames which are aborted during transmission are jammed. Such an interruption of transmission can be caused by any reason indicated by any of the status bits 8, 9, 10 and 12.

Jamming Rules

1. Jamming will not start before completion of preamble transmission.
2. Collisions detected during transmission of the last 11 bits will not result in jamming.

The format of a Transmit Buffer Descriptor is:

82586 Mode

31 ODD WORD 16	15	13	EVEN WORD 0	
NEXT TBD OFFSET	EOF	X	SIZE (ACT COUNT)	0
X X X X X X X X	TRANSMIT BUFFER ADDRESS			4

32-Bit Segmented Mode

31 ODD WORD 16	15	13	EVEN WORD 0	
NEXT TBD OFFSET	EOF	0	SIZE (ACT COUNT)	0
TRANSMIT BUFFER ADDRESS				4

Linear Mode

31 ODD WORD 16	15	13	EVEN WORD 0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	EOF	0	SIZE (ACT COUNT)	0
NEXT TBD ADDRESS				4
TRANSMIT BUFFER ADDRESS				8

Figure 31

where:

EOF — This bit indicates that this TBD is the last one associated with the frame being transmitted. It is set by the CPU before transmit.

SIZE (ACT COUNT) — This 14-bit quantity specifies the number of bytes that hold information for the current buffer. It is set by the CPU before transmission.

NEXT TBD ADDRESS — In the 82586 and 32-bit Segmented modes, it is the offset of the next TBD on the list. In the Linear mode this is the 32-bit address of the next TBD on the list. It is meaningless if EOF = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the data to be sent. In the 82586 mode, this is a 24-bit address (A31–A24 are considered to be zero). In the 32-bit Segmented and Linear modes this is a 32-bit address.

TDR

This operation activates Time Domain Reflectomet, which is a mechanism to detect open or short circuits on the link and their distance from the diagnosing station. The TDR command has no parameters. The TDR transmit sequence was changed, compared to the 82586, to form a regular transmission. The TDR bit stream is as follows.

— Preamble

— Source address

— Another Source address (the TDR frame is transmitted back to the sending station, so DEST ADR = SRC ADR).

— Data field containing 7Eh patterns.

— Jam Pattern, which is the inverse CRC of the transmitted frame.

Maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal. The following table lists the possible cases that the 82596 is able to analyze.

Conditions of TDR as Interpreted by the 82596

Transceiver Type / Condition	Ethernet	Non Ethernet
Carrier Sense was inactive for 2048-bit-time periods	Short or Open on the Transceiver Cable	NA
Carrier Sense signal dropped	Short on the Ethernet cable	NA
Collision Detect went active	Open on the Ethernet cable	Open on the Serial Link
The Carrier Sense Signal did not drop or the Collision Detect did not go active within 2048-bit time period	No Problem	No Problem

An Ethernet transceiver is defined as one that returns transmitted data on the receive pair and activates the Carrier Sense Signal while transmitting. A Non-Ethernet Transceiver is defined as one that does not do so.

The format of the Time Domain Reflectometer command is:

82586 and 32-Bit Segmented Modes

31 ODD WORD 16	15 EVEN WORD 0
EL, S, I, X X X X X X X X X X, 1 0 1, C, B, OK, 0 0 0 0 0 0 0 0 0 0 0 0 0	
LNK OK, XVR PRB, ET OPN, ET SRT, X, TIME (11 bits)	A15 LINK OFFSET A0

Linear Mode

31 ODD WORD 16	15 EVEN WORD 0
EL, S, I, 0 0 0 0 0 0 0 0 0 0 0, 1 0 1, C, B, OK, 0 0 0 0 0 0 0 0 0 0 0 0 0	
A31 LINK ADDRESS A0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	LNK OK, XVR PRB, ET OPN, ET SRT, X, TIME (11 bits)

Figure 32. TDR

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).

CMD (Bits 16–18) — The TDR command. Value: 5h.

TIME — An 11-bit field that specifies the number of TxC cycles that elapsed before an echo was observed. No echo is indicated by a reception consisting of "1s" only. Because the network contains various elements such as transceiver links, transceivers, Ethernet, repeaters etc., the TIME is not exactly proportional to the problems distance.

LNK OK (Bit 15) — No link problem identified. TIME = 7FFh.

XCVR PRB (Bit 14) — Indicates a Transceiver problem. Carrier Sense was inactive for 2048-bit time period. LNK OK = 0. TIME = 7FFh.

ET OPN (Bit 13) — The transmission line is not properly terminated. Collision Detect went active and LNK OK = 0.

ET SRT (Bit 12) — There is a short circuit on the transmission line. Carrier Sense Signal dropped and LNK OK = 0.

DUMP

This command causes the contents of various 82596 registers to be placed in a memory area specified by the user. It is supplied as a 82596 self-diagnostic tool, and to provide registers of interest to the user. The format of the DUMP command is:

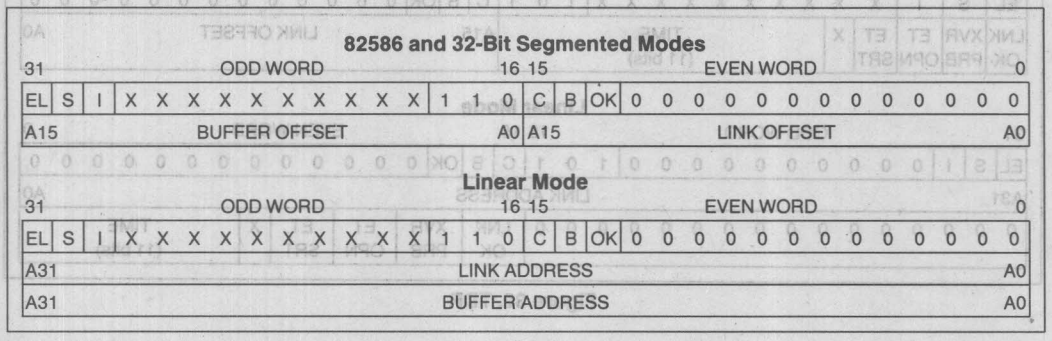


Figure 33. Dump

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).

OK — Indicates error free completion.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).

CMD (Bits 16–18) — The Dump command. Value: 6h.

BUFFER POINTER — In the 82586 and 32-bit Segmented modes this is the 16-bit-offset portion of the dump area address. In the Linear mode this is the 32-bit linear address of the dump area.

Dump Area Information Format

- The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
- In 82586 mode the dump area is 170 bytes.
- The DUMP area format of the 32-bit Segmented and Linear modes is described in Figure 35.
- The size of the dump area of the 32-bit Segmented and Linear modes is 304 bytes.
- When the Dump is executed by the Port command an extra word will be appended to the Dump Area. The extra word is a copy of the Dump Area status word (containing the C, B, and OK Bits). The C and OK Bits are set when the 82596 has completed the Port Dump command.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
DMA CONTROL REGISTER	00
CONFIGURE BYTES* 3, 2	02
CONFIGURE BYTES* 5, 4	04
CONFIGURE BYTES* 7, 6	06
CONFIGURE BYTES* 9, 8	08
CONFIGURE BYTES* 10	0A
I.A. BYTES 1, 0*	0C
I.A. BYTES 3, 2*	0E
I.A. BYTES 5, 4*	10
LAST T.X. STATUS*	12
T.X. CRC BYTES 1, 0*	14
T.X. CRC BYTES 3, 2*	16
R.X. CRC BYTES 1, 0*	18
R.X. CRC BYTES 3, 2*	1A
R.X. TEMP MEMORY 1, 0*	1C
R.X. TEMP MEMORY 3, 2*	1E
R.X. TEMP MEMORY 5, 4*	20
LAST RECEIVED STATUS*	22
HASH REGISTER BYTES 1, 0*	24
HASH REGISTER BYTES 3, 2*	26
HASH REGISTER BYTES 5, 4*	28
HASH REGISTER BYTES 7, 6*	2A
SLOT TIME COUNTER*	2C
WAIT TIME COUNTER*	2E
MICRO MACHINE**	30
REGISTER FILE	.
60 BYTES	6A
MICRO MACHINE LFSR**	6C
MICRO MACHINE**	6E
FLAG ARRAY	.
14 BYTES	7A
QUEUE MEMORY**	7C
CU PORT	.
8 BYTES	82
MICRO MACHINE ALU**	84
RESERVED**	86
M.M. TEMP A ROTATE R**	88
M.M. TEMP A**	8A
T.X. DMA BYTE COUNT**	8C
M.M. INPUT PORT ADDRESS**	8E
T.X. DMA ADDRESS	90
M.M. OUTPUT PORT**	92
R.X. DMA BYTE COUNT**	94
R.U. DMA ADDRESS**	96
M.M. OUTPUT PORT ADDRESS REGISTER**	98
RESERVED**	9A
BUS THROTTLE TIMERS	9C
DIU CONTROL REGISTER**	9E
RESERVED**	A0
DMA CONTROL REGISTER**	A2
BIU CONTROL REGISTER**	A4
M.M. DISPATCHER REG.**	A6
M.M. STATUS REGISTER**	A8

*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586.

**These bytes are not user defined, results may vary from Dump command.

Figure 34. Dump Area Format—82586 Mode

31	0
CONFIGURE BYTES 5, 4, 3, 2	00
CONFIGURE BYTES 9, 8, 7, 6	04
CONFIGURE BYTES 13, 12, 11, 10	08
I.A. BYTES 1, 0 | X X X X X X X X	0C
I.A. BYTES 5, 2	10
TX CRC BYTES 0, 1 | LAST T.X. STATUS	14
RX CRC BYTES 0, 1 | TX CRC BYTES 3, 2	18
RX TEMP MEMORY 1, 0 | RX CRC BYTES 3, 2	1C
R.X. TEMP MEMORY 5, 2	20
HASH REGISTERS 1, 0 | LAST R.X. STATUS	24
HASH REGISTER BYTES 5, 2	28
SLOT TIME COUNTER | HASH REGISTERS 7, 6	2C
RECEIVE FRAME LENGTH | WAIT-TIME COUNTER	30
MICRO MACHINE** REGISTER FILE 128 BYTES	34 . . . B0
MICRO MACHINE LFSR**	B4
MICRO MACHINE** FLAG ARRAY 28 BYTES	B8 . . . D0
M.M. INPUT PORT** 16 BYTES	D4 . E0
MICRO MACHINE ALU**	E4
RESERVED**	E8
M.M. TEMP A ROTATE R.**	EC
M.M. TEMP A**	F0
T.X. DMA BYTE COUNT**	F4
M.M. INPUT PORT ADDRESS REGISTER**	F8
T.X. DMA ADDRESS**	FC
M.M. OUTPUT PORT REGISTER**	100
R.X. DMA BYTE COUNT**	104
M.M. OUTPUT PORT ADDRESS REGISTER**	108
R.X. DMA ADDRESS REGISTER**	10C
RESERVED**	110
BUS THROTTLE TIMERS	114
DIU CONTROL REGISTER**	118
RESERVED**	11C
DMA CONTROL REGISTER**	120
BIU CONTROL REGISTER**	124
M.M. DISPATCHER REG.**	128
M.M. STATUS REGISTER**	12C

*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586.

**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 35. Dump Area Format—Linear and 32-Bit Segmented Mode

Diagnose

The Diagnose Command triggers an internal self-test procedure that checks internal 82596 hardware, which includes:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter.
- Slot Time Period Counter.
- Collision Number Counter.
- Exponential Backoff Shift Register.
- Exponential Backoff Mask Logic.
- Timer Trigger Logic.

This procedure checks the operation of the Backoff block, which resides in the serial side and is not easily controlled. The Diagnose command is performed in two phases.

The format of the 82596 Diagnose command is:

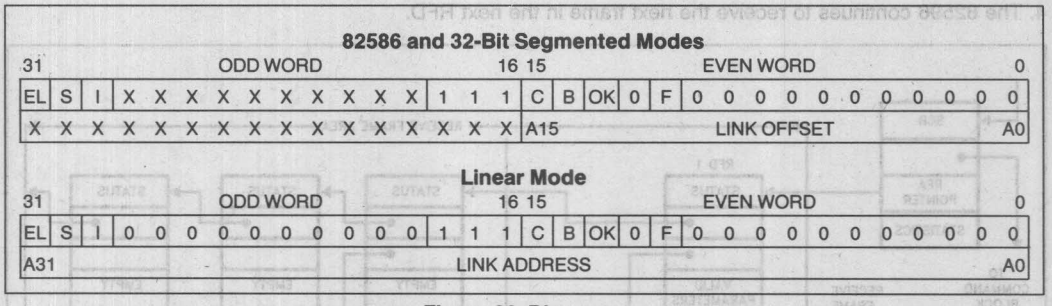


Figure 36. Diagnose

where:

LINK ADDRESS, EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear Modes).
CMD (bits 16–18)	— The Diagnose command. Value: 7h.
OK (bit 13)	— Indicates error free completion.
F (bit 11)	— Indicates that the self-test procedure has failed.

RECEIVE FRAME DESCRIPTOR

Each received frame is described by one Receive Frame Descriptor (see Figure 37). Two new memory structures are available for the received frames. The structures are available only in the Linear and 32-bit Segmented modes.

Simplified Memory Structure

The first is the Simplified memory structure, the data section of the received frame is part of the RFD and is located immediately after the Length Field. Receive Buffer Descriptors are not used with the Simplified structure, it is primarily used to make programming easier. If the length of the data area described in the Size Field is smaller than the incoming frame, the following happens.

1. The received frame is truncated.
2. The No Resource error counter is updated.
3. If the 82596 is configured to Save Bad Frames the RFD is not reused; otherwise, the same RFD is used to hold the next received frame, and the only action taken regarding the truncated frame is to update the counter.
4. The 82596 continues to receive the next frame in the next RFD.

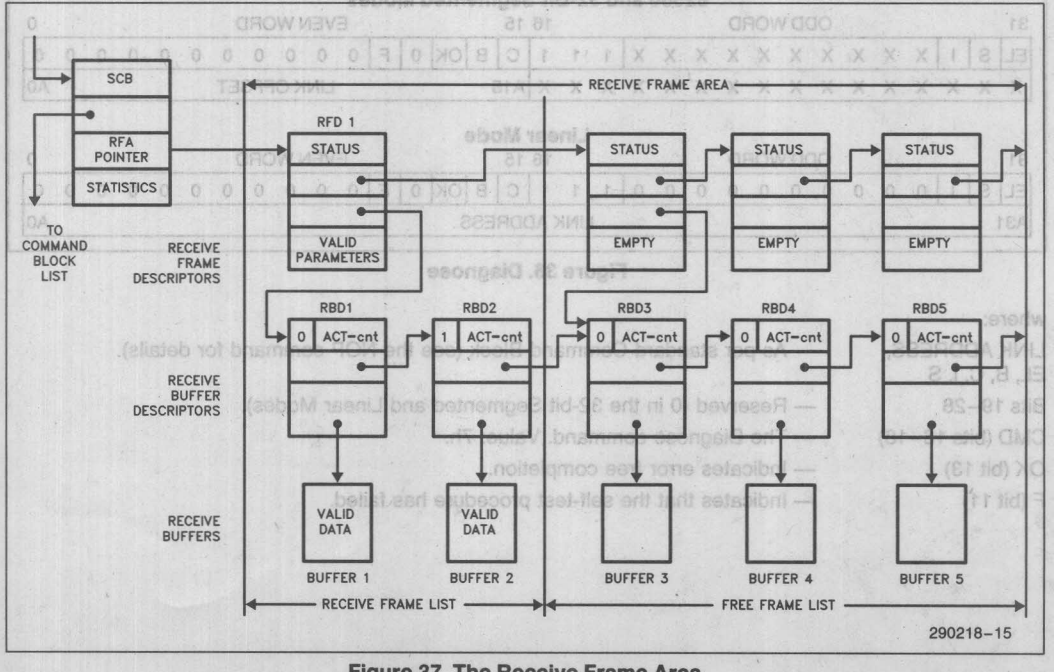


Figure 37. The Receive Frame Area

Note that this sequence is very useful for monitoring. If the 82596 is configured to Save Bad Frames, to receive in Promiscuous mode, and to use the Simplified memory structure, any programmed length of received data can be saved in memory.

The Simplified memory structure is shown in Figure 38.

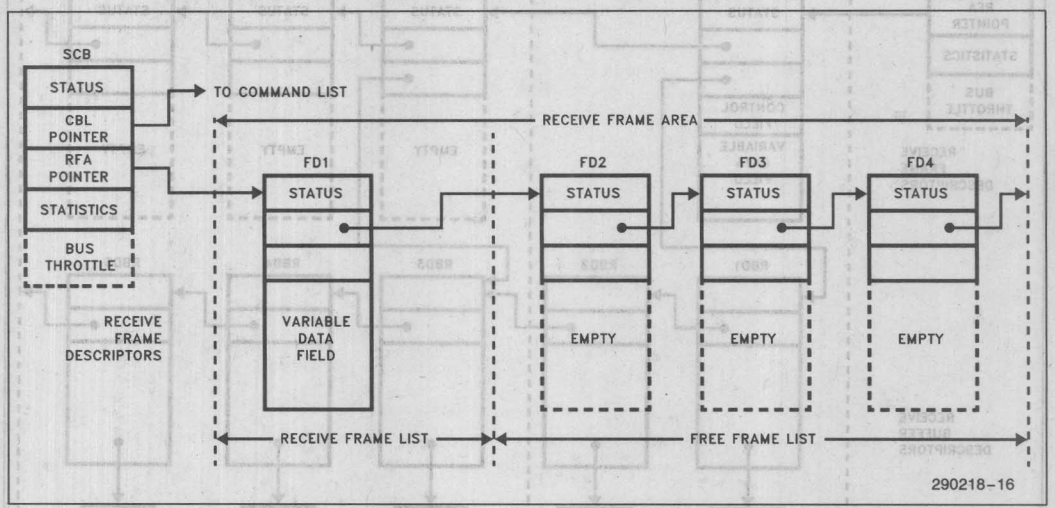


Figure 38. RFA Simplified Memory Structure

Flexible Memory Structure

The second structure is the Flexible memory structure, the data structure of the received frame is stored in both the RFD and in a linked list of Receive Buffers—Receive Buffer Descriptors. The received frame is placed in the RFD as configured in the Size field. Any remaining data is placed in a linked list of RBDs.

The Flexible memory structure is shown in Figure 39.

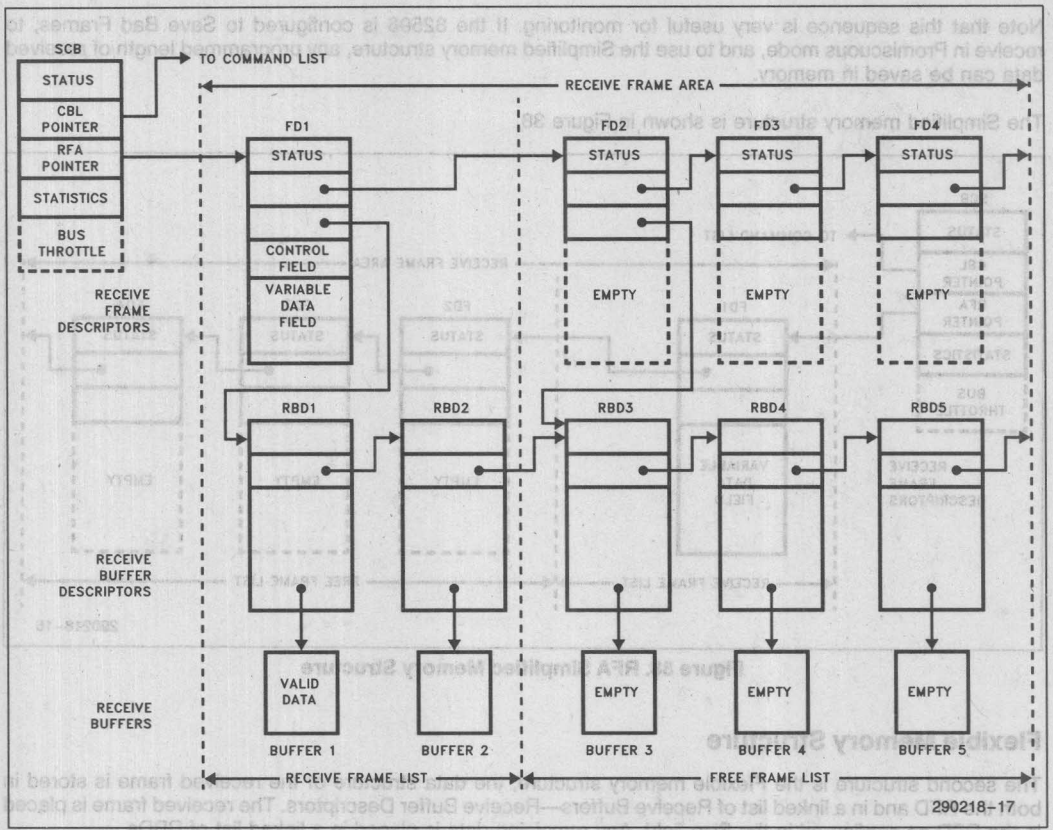


Figure 39. RFA Flexible Memory Structure

Buffers on the receive side can be different lengths. The 82596 will not place more bytes into a buffer than indicated in the associated RBD. The 82596 will fetch the next RBD before it is needed. The 82596 will attempt to receive frames as long as the FBL is not exhausted. If there are no more buffers, the 82596 Receive Unit will enter the No Resources state. Before starting the RU, the CPU must place the FBL pointer in the RBD pointer field of the first RFD. All remaining RBD pointer fields for subsequent RFDs should be "1s." If the Receive Frame Descriptor and the associated Receive Buffers are not reused (e.g., the frame is properly received or the 82596 is configured to Save Bad Frames), the 82596 writes the address of the next free RBD to the RBD pointer field of the next RFD.

Receive Buffer Descriptor (RBD)

The RBDs are used to store received data in a flexible set of linked buffers. The portion of the frame's data field that is outside the RFD is placed in a set of buffers chained by a sequence of RBDs. The RFD points to the first RBD, and the last RBD is flagged with an EOF bit set to 1. Each buffer in the linked list of buffers related to a particular frame can be any size up to 2^{14} bytes but must be word aligned (begin on an even numbered byte). This ensures optimum use of the memory resources while maintaining low overhead. All buffers in a frame are filled with the received data except for the last, in which the actual count can be smaller than the allocated buffer space.

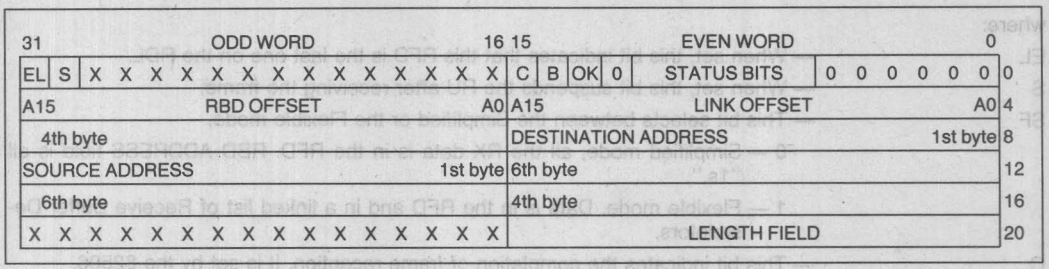


Figure 40. Receive Frame Descriptor—82586 Mode

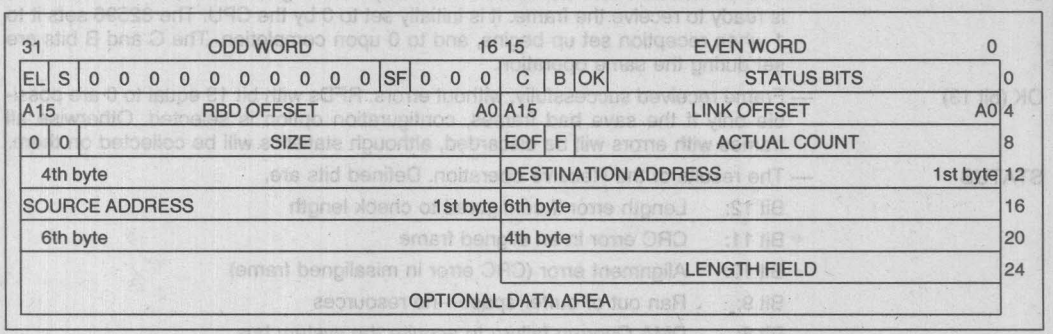


Figure 41. Receive Frame Descriptor—32-Bit Segmented Mode

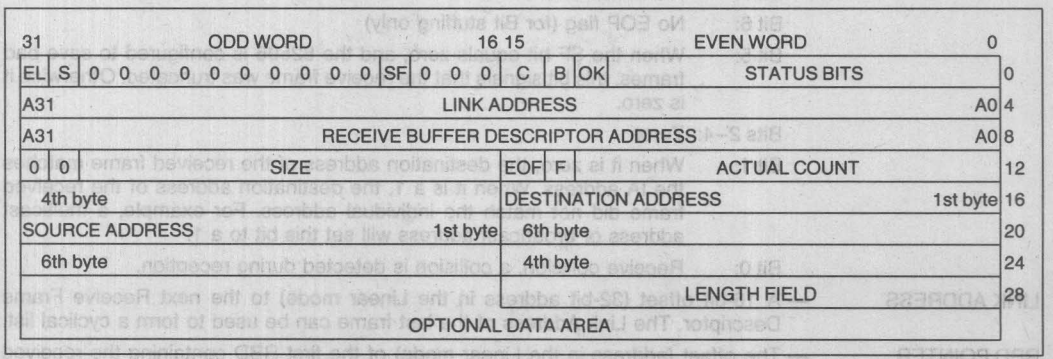


Figure 42. Receive Frame Descriptor—Linear Mode

where:

EL — When set, this bit indicates that this RFD is the last one on the RDL.

S — When set, this bit suspends the RU after receiving the frame.

SF — This bit selects between the Simplified or the Flexible mode.

0 — Simplified mode, all the RX data is in the RFD. RBD ADDRESS field is all "1s."

1 — Flexible mode. Data is in the RFD and in a linked list of Receive Buffer Descriptors.

C — This bit indicates the completion of frame reception. It is set by the 82596.

B — This bit indicates that the 82596 is currently receiving this frame, or that the 82596 is ready to receive the frame. It is initially set to 0 by the CPU. The 82596 sets it to 1 when reception set up begins, and to 0 upon completion. The C and B bits are set during the same operation.

OK (bit 13) — Frame received successfully, without errors. RFDs with bit 13 equal to 0 are possible only if the save bad frames, configuration option is selected. Otherwise all frames with errors will be discarded, although statistics will be collected on them.

STATUS — The results of the Receive operation. Defined bits are,

Bit 12:	Length error if configured to check length
Bit 11:	CRC error in an aligned frame
Bit 10:	Alignment error (CRC error in misaligned frame)
Bit 9:	Ran out of buffer space—no resources
Bit 8:	DMA Overrun failure to acquire the system bus.
Bit 7:	Frame too short.
Bit 6:	No EOP flag (for Bit stuffing only)
Bit 5:	When the SF bit equals zero, and the 82596 is configured to save bad frames, this bit signals that the receive frame was truncated. Otherwise it is zero.
Bits 2–4:	Zeros
Bit 1:	When it is zero, the destination address of the received frame matches the IA address. When it is a 1, the destination address of the received frame did not match the individual address. For example, a multicast address or broadcast address will set this bit to a 1.
Bit 0:	Receive collision, a collision is detected during reception.

LINK ADDRESS — A 16-bit offset (32-bit address in the Linear mode) to the next Receive Frame Descriptor. The Link Address of the last frame can be used to form a cyclical list.

RBD POINTER — The offset (address in the Linear mode) of the first RBD containing the received frame data. An RBD pointer of all ones indicates no RBD.

EOF
F
SIZE
ACT COUNT — These fields are for the Simplified and Flexible memory models. They are exactly the same as the respective fields in the Receive Buffer Descriptor. See the next section for detailed explanation of their functions.

MC — Multicast bit.

DESTINATION ADDRESS — The contents of the destination address of the receive frame. The field is 0 to 6 bytes long.

SOURCE ADDRESS — The contents of the Source Address field of the received frame. It is 0 to 6 bytes long.

LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is received, i.e., most significant byte first, least significant byte second.

NOTES

1. The Destination address, Source address and Length fields are packed, i.e., one field immediately follows the next.
2. The affect of Address/Length Location (No Source Address Insertion) configuration parameter while receiving is as follows:

— 82586 Mode: The Destination address, Source address and Length field are not used, they are placed in the RX data buffers.

— 32-Bit Segmented and Linear Modes: when the Simplified memory model is used, the Destination address, Source address and Length fields reside in their respective fields in the RFD. When the Flexible memory strucrture is used the Destination address, Source address, and Length field locations depend on the SIZE field of the RFD. They can be placed in the RFD, in the RX data buffers, or partially in the RFD and the rest in the RX data buffers, depending on the SIZE field value.

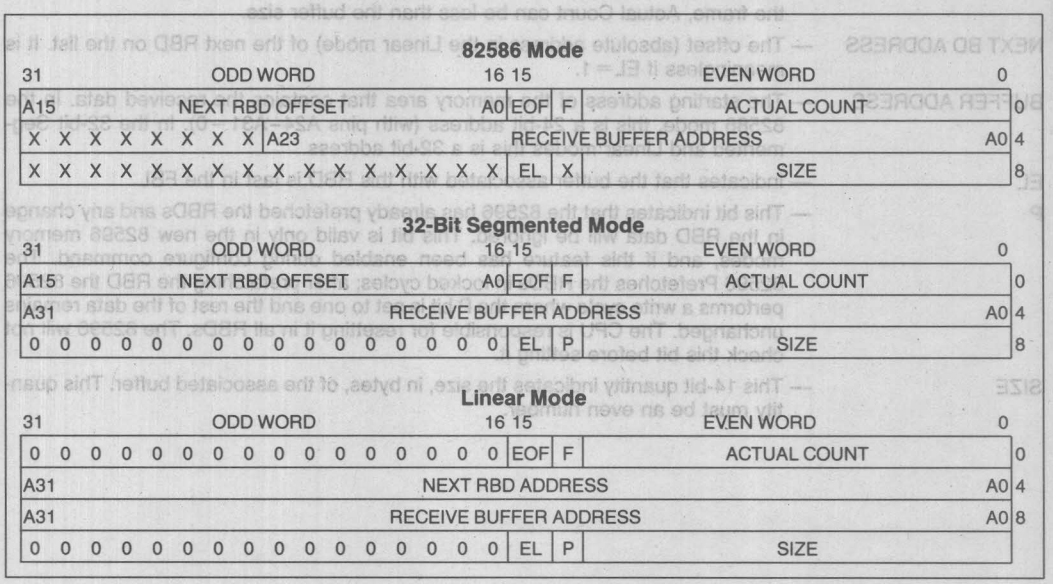


Figure 43. Receive Buffer Descriptor

where:

EOF — Indicates that this is the last buffer related to the frame. It is cleared by the CPU before starting the RU, and is written by the 82596 at the end of reception of the frame.

F — Indicates that this buffer has already been used. The Actual Count has no meaning unless the F bit equals one. This bit is cleared by the CPU before starting the RU, and is set by the 82596 after the associated buffer has been. This bit has the same meaning as the Complete bit in the RFD and CB.

ACT COUNT — This 14-bit quantity indicates the number of meaningful bytes in the buffer. It is cleared by the CPU before starting the RU, and is written by the 82596 after the associated buffer has already been used. In general, after the buffer is full, the Actual Count value equals the size field of the same buffer. For the last buffer of the frame, Actual Count can be less than the buffer size.

NEXT BD ADDRESS — The offset (absolute address in the Linear mode) of the next RBD on the list. It is meaningless if EL = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the received data. In the 82586 mode, this is a 24-bit address (with pins A24–A31 = 0). In the 32-bit Segmented and Linear modes this is a 32-bit address.

EL — Indicates that the buffer associated with this RBD is last in the FBL.

P — This bit indicates that the 82596 has already prefetched the RBDs and any change in the RBD data will be ignored. This bit is valid only in the new 82596 memory modes, and if this feature has been enabled during configure command. The 82596 Prefetches the RBDs in locked cycles; after prefetching the RBD the 82596 performs a write cycle where the P bit is set to one and the rest of the data remains unchanged. The CPU is responsible for resetting it in all RBDs. The 82596 will not check this bit before setting it.

SIZE — This 14-bit quantity indicates the size, in bytes, of the associated buffer. This quantity must be an even number.

ELECTRICAL AND TIMING CHARACTERISTICS

DC Characteristics

T_C = 0°C–85°C, V_{CC} = 5V ±10% LE/$\overline{BE}$ have MOS levels (see V_{MIL}, V_{MIH}).
All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, V_{OH}).

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage (TTL)	−0.3	+0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.3	V	
V_{MIL}	Input Low Voltage (MOS)	−0.3	+0.8	V	
V_{MIH}	Input High Voltage (MOS)	3.7	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 4.0 mA(1)
V_{CIL}	$\overline{RXC}$, $\overline{TXC}$ Input Low Voltage	−0.5	0.6	V	
V_{CIH}	$\overline{RXC}$, $\overline{TXC}$ Input High Voltage	3.3	V_{CC}+0.5	V	
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} = 0.9mA–1 mA(1)
I_{LI}	Input Leakage Current		±15	μA	0 ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±15	μA	0.45 < V_{OUT} < V_{CC}
C_{IN}	Capacitance of Input Buffer		10	pF	FC = 1 MHz
C_{OUT}	Capacitance of Input/Output Buffer		12	pF	FC = 1 MHz
C_{CLK}	CLK Capacitance		20	pF	FC = 1 MHz
I_{CC}	Power Supply		200	mA	At 25 MHz
I_{CC}	Power Supply		300	mA	At 33 MHz

SEE INTEL PRELIMINARY DATASHEET FOR SUPPLEMENT

AC Characteristics

82596CA INPUT/OUTPUT SYSTEM TIMINGS

T_C = 0°C–85°C, V_{CC} = 5V ±10%. These timing assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF however timings must be derated. All timing requirements are given in nanoseconds.

Symbol	Parameter	25 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	25 MHz	1X CLK Input
T1	CLK Period	40	80	
T1a	CLK Period Stability		0.1%	Adjacent CLK Δ
T2	CLK High	14		2.0V
T3	CLK Low	14		0.8V
T4	CLK Rise Time		4	0.8V to 2.0V
T5	CLK Fall Time		4	2.0V to 0.8V
T6	$\overline{BEn}$, $\overline{LOCK}$, and A2–A31 Valid Delay	3	22	
T6a	$\overline{BLAST}$, $\overline{PCHK}$ Valid Delay	3	27	
T7	$\overline{BEn}$, $\overline{LOCK}$, $\overline{BLAST}$, A2–A31 Float Delay	3	30	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	3	22	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	3	30	
T10	D0–D31, DPn Write Data Valid Delay	3	22	
T11	D0–D31, DPn Write Data Float Delay	3	30	
T12	HOLD Valid Delay	3	22	
T13	CA and BREQ Setup Time	7		1, 2
T14	CA and BREQ Hold Time	3		1, 2
T15	$\overline{BS16}$ Setup Time	8		2
T16	$\overline{BS16}$ Hold Time	3		2
T17	$\overline{BRDY}$, $\overline{RDY}$ Setup Time	8		2
T18	$\overline{BRDY}$, $\overline{RDY}$ Hold Time	3		2
T19	D0–D31, DPn READ Setup Time	5		2
T20	D0–D31, DPn READ Hold Time	3		2
T21	AHOLD and HLDA Setup Time	10		1, 2
T22	AHOLD Hold Time	3		1, 2
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	10		1, 2
T24	RESET Hold Time	3		1, 2
T25	INT/$\overline{INT}$ Valid Delay	1	26	
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	2 T1		1, 2, 3
T27	D0–D31 CPU $\overline{PORT}$ Access Setup Time	5		2
T28	D0–D31 CPU $\overline{PORT}$ Access Hold Time	3		2
T29	$\overline{PORT}$ Setup Time	7		2
T30	$\overline{PORT}$ Hold Time	3		2
T31	$\overline{BOFF}$ Setup Time	10		2
T32	$\overline{BOFF}$ Hold Time	3		2

SEE PRELIMINARY INTEL 82596CA DATASHEET SUPPLEMENT

AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

T_C = 0°C–85°C, V_{CC} = 5V ±5%. These timing assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however timings must be derated. All timing requirements are given in nanoseconds.

Symbol	Parameter	33 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	33 MHz	1X CLK Input
T1	CLK Period	30	80	
T1a	CLK Period Stability		0.1%	Adjacent CLK Δ
T2	CLK High	11		2.0V
T3	CLK Low	11		0.8V
T4	CLK Rise Time		3	0.8V to 2.0V
T5	CLK Fall Time		3	2.0V to 0.8V
T6	$\overline{BEn}$, $\overline{LOCK}$, and A2–A31 Valid Delay	3	19	
T6a	$\overline{BLAST}$, $\overline{PCHK}$ Valid Delay	3	22	
T7	$\overline{BEn}$, $\overline{LOCK}$, $\overline{BLAST}$, A2–A31 Float Delay	3	20	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	3	19	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	3	20	
T10	D0–D31, DPn Write Data Valid Delay	3	19	
T11	D0–D31, DPn Write Data Float Delay	3	20	
T12	HOLD Valid Delay	3	19	
T13	CA and $\overline{BREQ}$ Setup Time	7		1, 2
T14	CA and $\overline{BREQ}$ Hold Time	3		1, 2
T15	$\overline{BS16}$ Setup Time	6		2
T16	$\overline{BS16}$ Hold Time	3		2
T17	$\overline{BRDY}$, $\overline{RDY}$ Setup Time	6		2
T18	$\overline{BRDY}$, $\overline{RDY}$ Hold Time	3		2
T19	D0–D31, DPn READ Setup Time	5		2
T20	D0–D31, DPn READ Hold Time	3		2
T21	AHOLD and HLDA Setup Time	8		1, 2
T22	AHOLD Hold Time	3		1, 2

PRELIMINARY DATASHEET
SEE INTEL FOR DATASHEET SUPPLEMENT

AC Characteristics (Continued)

82596CA INPUT/OUTPUT SYSTEM TIMINGS

C_L on all outputs is 50 pF unless otherwise specified.
All timing requirements are given in nanoseconds.

Symbol	Parameter	33 MHz		Notes
		Min	Max	
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	8		1, 2
T24	RESET Hold Time	3		1, 2
T25	INT/$\overline{\text{INT}}$ Valid Delay	1	20	
T26	CA and BREQ, $\overline{\text{PORT}}$ Pulse Width	2T1		1, 2, 3
T27	D0–D31 CPU $\overline{\text{PORT}}$ Access Setup Time	5		2
T28	D0–D31 CPU $\overline{\text{PORT}}$ Access Hold Time	3		2
T29	$\overline{\text{PORT}}$ Setup Time	7		2
T30	$\overline{\text{PORT}}$ Hold Time	3		2
T31	$\overline{\text{BOFF}}$ Setup Time	8		2
T32	$\overline{\text{BOFF}}$ Hold Time	3		2

PRELIMINARY SEE INTEL FOR DATASHEET SUPPLEMENT

NOTES:
1. RESET, HLDA, and CA are internally synchronized. This timing is to guarantee recognition at next clock for RESET, HLDA and CA.
2. All set-up, hold and delay timings are at maximum frequency specification Fmax, and must be derated according to the following equation for operation at lower frequencies:
   Tderated = (Fmax/Fopr) × T
   where:
   Tderate = Specifies the value to derate the specification.
   Fmax = Maximum operating frequency.
   Fopr = Actual operating frequency.
   T = Specification at maximum frequency.
   This calculation only provides a rough estimate for derating the frequency. For more detailed information, contact your Intel Sales Office for the data sheet supplement.
3. CA pulse width need only be 1 T1 wide if the set up and hold times are met; BREQ must meet setup and hold times and need only be 1 T1 wide.

TRANSMIT/RECEIVE CLOCK PARAMETERS

Symbol	Parameter	20 MHz		Notes
		Min	Max	
T36	$\overline{\text{TxC}}$ Cycle	50		1, 3
T38	$\overline{\text{TxC}}$ Rise Time		5	1
T39	$\overline{\text{TxC}}$ Fall Time		5	1
T40	$\overline{\text{TxC}}$ High Time	19		1, 3
T41	$\overline{\text{TxC}}$ Low Time	18		1, 3
T42	TxD Rise Time		10	4
T43	TxD Fall Time		10	4
T44	TxD Transition	20		2, 4
T45	$\overline{\text{TxC}}$ Low to TxD Valid		25	4, 6
T46	$\overline{\text{TxC}}$ Low to TxD Transition		25	2, 4
T47	$\overline{\text{TxC}}$ High to TxD Transition		25	2, 4
T48	$\overline{\text{TxC}}$ Low to TxD High (At End of Transition)		25	4

PRELIMINARY SEE INTEL FOR DATASHEET SUPPLEMENT

TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

Symbol	Parameter	20 MHz		Notes
		Min	Max	
$\overline{RTS}$ AND $\overline{CTS}$ PARAMETERS				
T49	$\overline{TxC}$ Low to $\overline{RTS}$ Low, Time to Activate $\overline{RTS}$		25	5
T50	$\overline{CTS}$ Low to $\overline{TxC}$ Low, $\overline{CTS}$ Setup Time		20	
T51	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid, $\overline{CTS}$ Hold Time	10		7
T52	$\overline{TxC}$ Low to $\overline{RTS}$ High		25	5
RECEIVE CLOCK PARAMETERS				
T53	$\overline{RXC}$ Cycle	50		1, 3
T54	$\overline{RXC}$ Rise Time		5	1
T55	$\overline{RXC}$ Fall Time		5	1
T56	$\overline{RXC}$ High Time	19		1
T57	$\overline{RXC}$ Low Time	18		1
RECEIVED DATA PARAMETERS				
T58	RXD Setup Time	20		6
T59	RXD Hold Time	10		6
T60	RXD Rise Time		10	
T61	RXD Fall Time		10	
$\overline{CRS}$ AND $\overline{CDT}$ PARAMETERS				
T62	$\overline{CDT}$ Low to $\overline{TXC}$ HIGH External Collision Detect Setup Time	20		
T63	$\overline{TXC}$ High to $\overline{CDT}$ Inactive, $\overline{CDT}$ Hold Time	10		
T64	$\overline{CDT}$ Low to Jam Start			10
T65	$\overline{CRS}$ Low to $\overline{TXC}$ High, Carrier Sense Setup Time	20		
T66	$\overline{TXC}$ High to $\overline{CRS}$ Inactive, $\overline{CRS}$ Hold Time (Internal Collision Detect)	10		
T67	$\overline{CRS}$ High to Jamming Start,			12
T68	Jamming Period			11
T69	$\overline{CRS}$ High to $\overline{RXC}$ High, $\overline{CRS}$ Inactive Setup Time	30		
T70	$\overline{RXC}$ High to $\overline{CRS}$ High, $\overline{CRS}$ Inactive Hold Time	10		

PRELIMINARY SEE INTEL FOR DATASHEET SUPPLEMENT

TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

Symbol	Parameter	20 MHz		Notes
		Min	Max	
INTERFRAME SPACING PARAMETERS				
T71	Interframe Delay			9
EXTERNAL LOOPBACK-PIN PARAMETERS				
T72	$\overline{TXC}$ Low to $\overline{LPBK}$ Low		T36	4
T73	$\overline{TXC}$ Low to $\overline{LPBK}$ High		T36	4

PRELIMINARY
SEE INTEL FOR DATASHEET
SUPPLEMENT

NOTES:
1. Special MOS levels. V_{CIL} = 0.9V and V_{CIH} = 3.0V.
2. Manchester only.
3. Manchester. Needs 50% duty cycle.
4. 1 TTL load + 50 pF.
5. 1 TTL load + 100 pF.
6. NRZ only.
7. Abnormal end of transmission—CTS expires before RTS.
8. Normal end to transmission.
9. Programmable value:
   T71 = N_{IFS} • T36
   where: N_{IFS} = the IFS configuration value
   (if N_{IFS} is less than 12 then N_{IFS} is forced to 12).
10. Programmable value:
    T64 = (N_{CDF} • T36) + x • T36
    (If the collision occurs after the preamble)
    where:
    N_{CDF} = the collision detect filter configuration value, and
    x = 12, 13, 14, or 15
11. T68 = 32 • T36
12. Programmable value:
    T67 = (N_{CSF} • T36) + x • T36
    where: N_{CSF} = the Carrier Sense Filter configuration value, and
    x = 12, 13, 14, or 15
13. To guarantee recognition on the next clock.

82596CA BUS OPERATION

The following figures show the 82596CA basic bus cycle and basic burst cycle.

Please refer to the *32-Bit LAN Components Manual.*

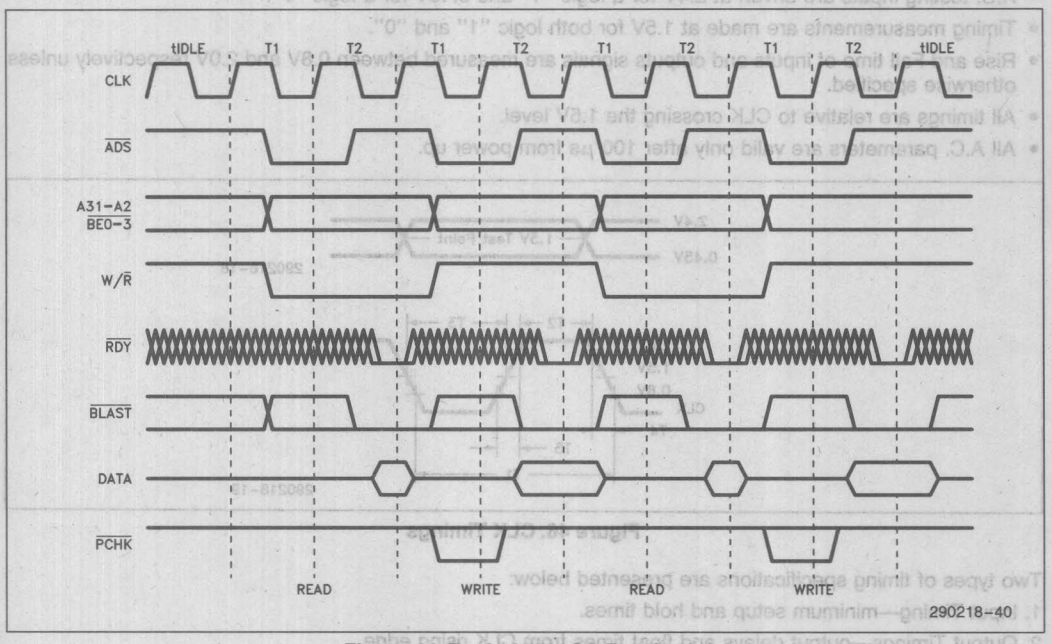


Figure 44. Basic 82596CA Bus Cycle

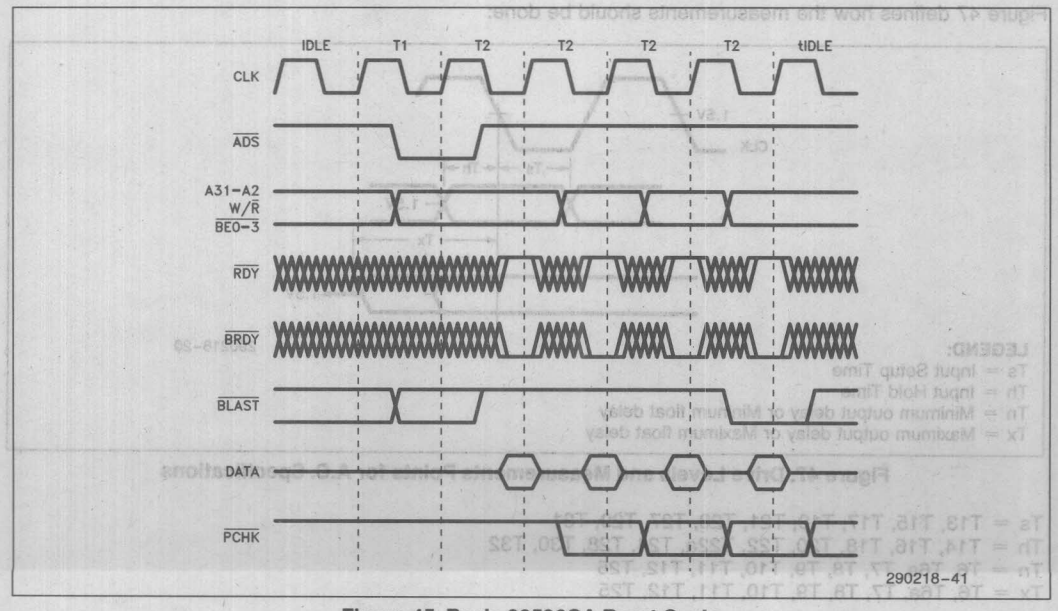


Figure 45. Basic 82596CA Burst Cycle

SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

The measurements should be done at:

- T_C = 0°C–85°C, V_{CC} = 5V ±10%, C = 50 pF unless otherwise specified.
- A.C. testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
- Timing measurements are made at 1.5V for both logic "1" and "0".
- Rise and Fall time of inputs and outputs signals are measured between 0.8V and 2.0V respectively unless otherwise specified.
- All timings are relative to CLK crossing the 1.5V level.
- All A.C. parameters are valid only after 100 μs from power up.

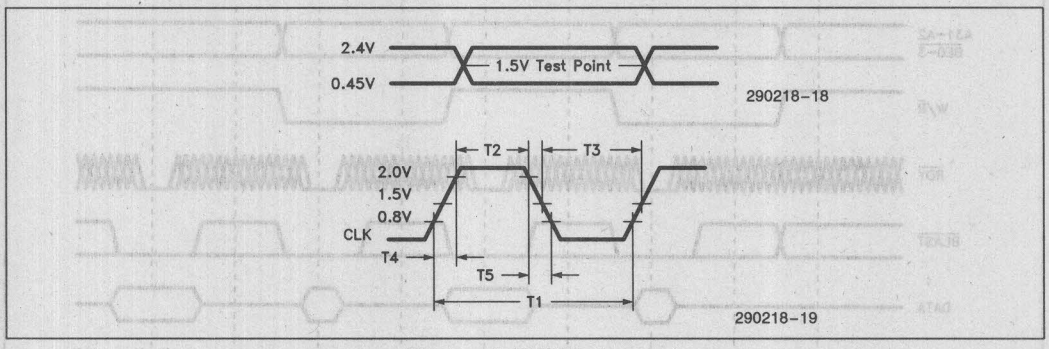
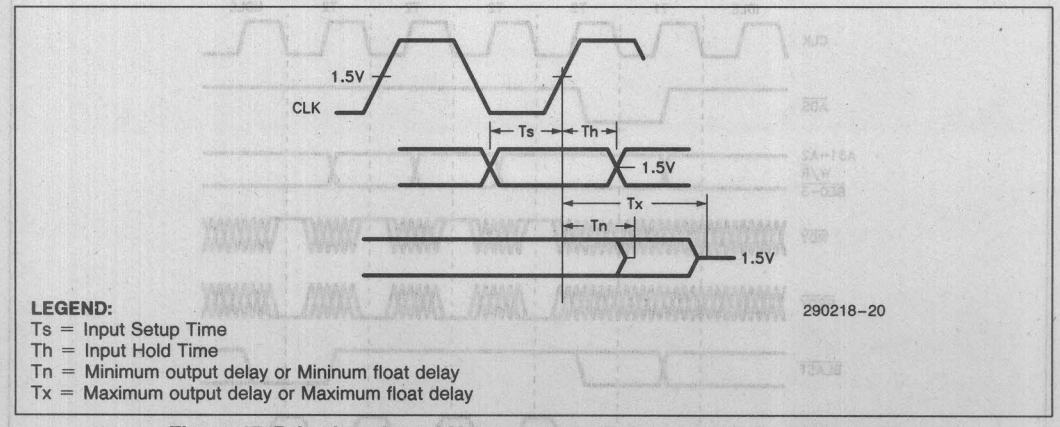


Figure 46. CLK Timings

Two types of timing specifications are presented below:

1. Input Timing—minimum setup and hold times.
2. Output Timings—output delays and float times from CLK rising edge.

Figure 47 defines how the measurements should be done:



LEGEND:
Ts = Input Setup Time
Th = Input Hold Time
Tn = Minimum output delay or Mininum float delay
Tx = Maximum output delay or Maximum float delay

Figure 47. Drive Levels and Measurements Points for A.C. Specifications

Ts = T13, T15, T17, T19, T21, T23, T27, T29, T31
Th = T14, T16, T18, T20, T22, T22a, T24, T28, T30, T32
Tn = T6, T6a, T7, T8, T9, T10, T11, T12, T25
Tx = T6, T6a, T7, T8, T9, T10, T11, T12, T25

INPUT WAVEFORMS

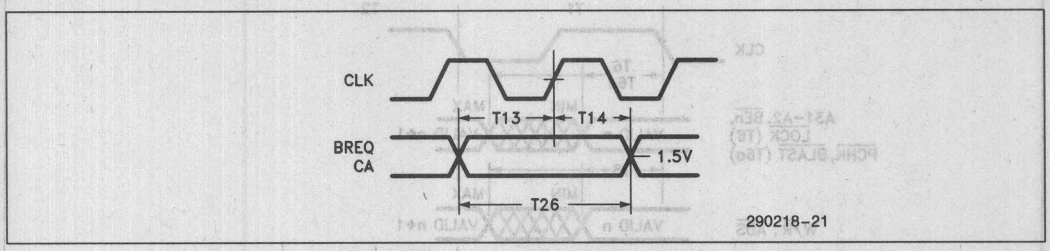


Figure 48. CA and BREQ Input Timing

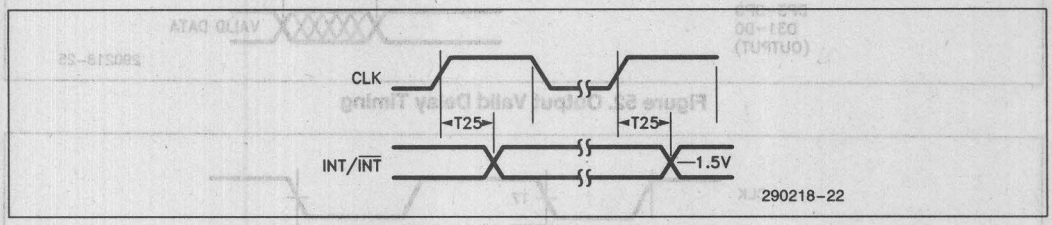


Figure 49. INT/INT Output Timing

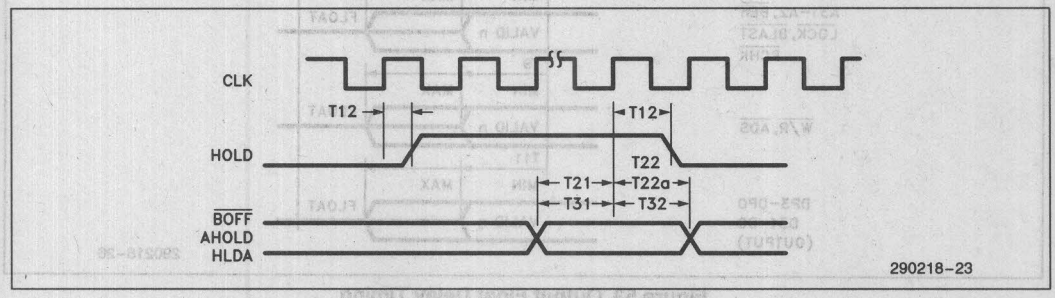


Figure 50. HOLD/HLDA Timings

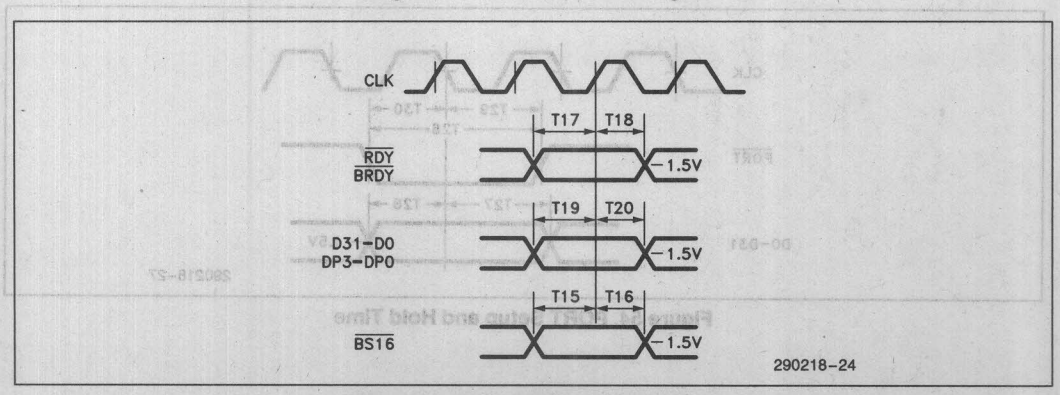


Figure 51. Input Setup and Hold Time

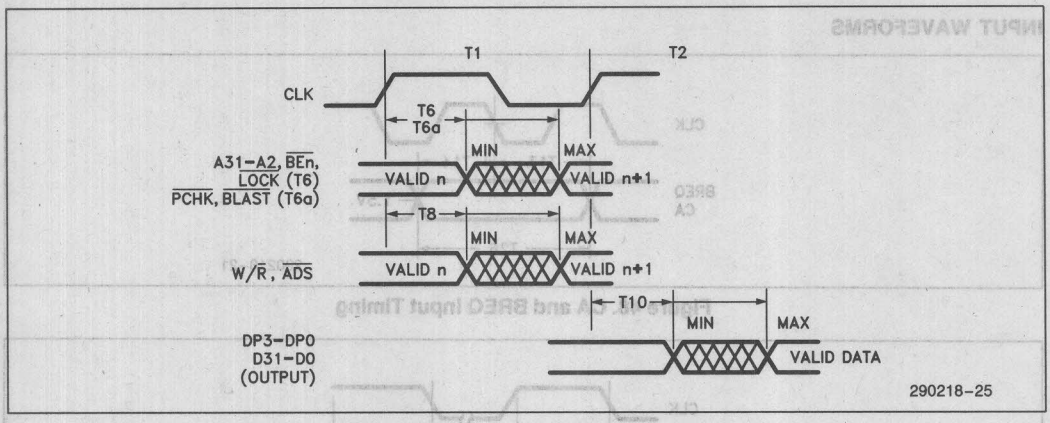


Figure 52. Output Valid Delay Timing

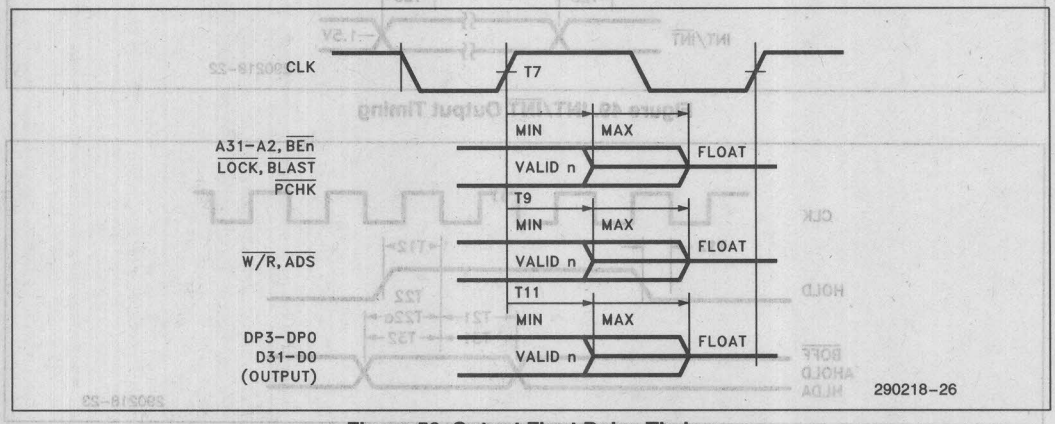


Figure 53. Output Float Delay Timing

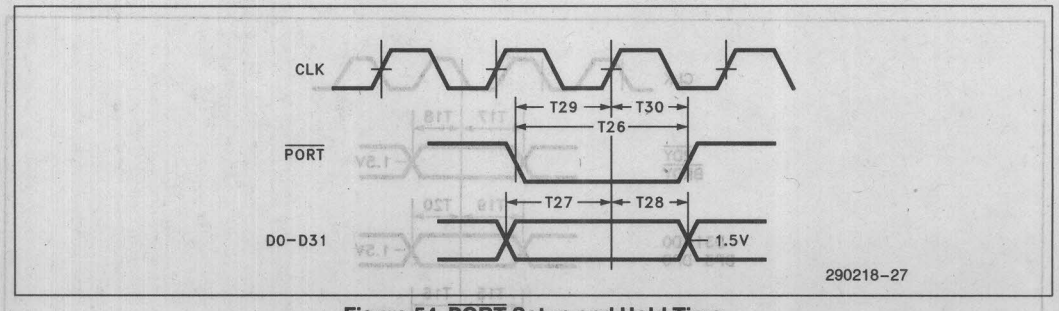


Figure 54. PORT Setup and Hold Time

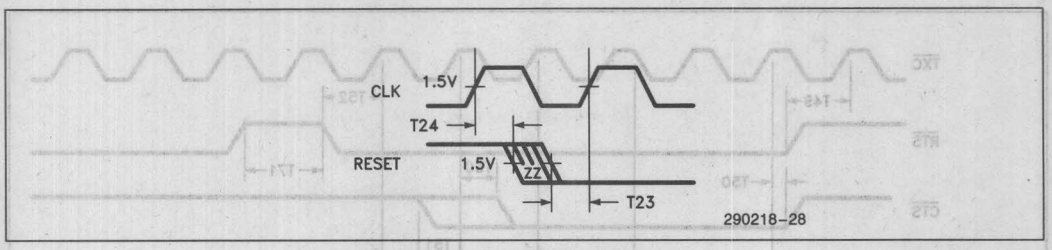


Figure 55. RESET Input Timing

SERIAL AC TIMING CHARACTERISTICS

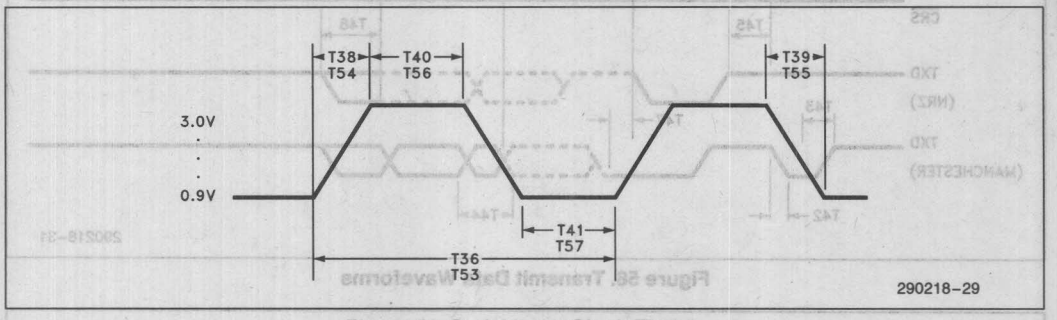


Figure 56. Serial Input Clock Timing

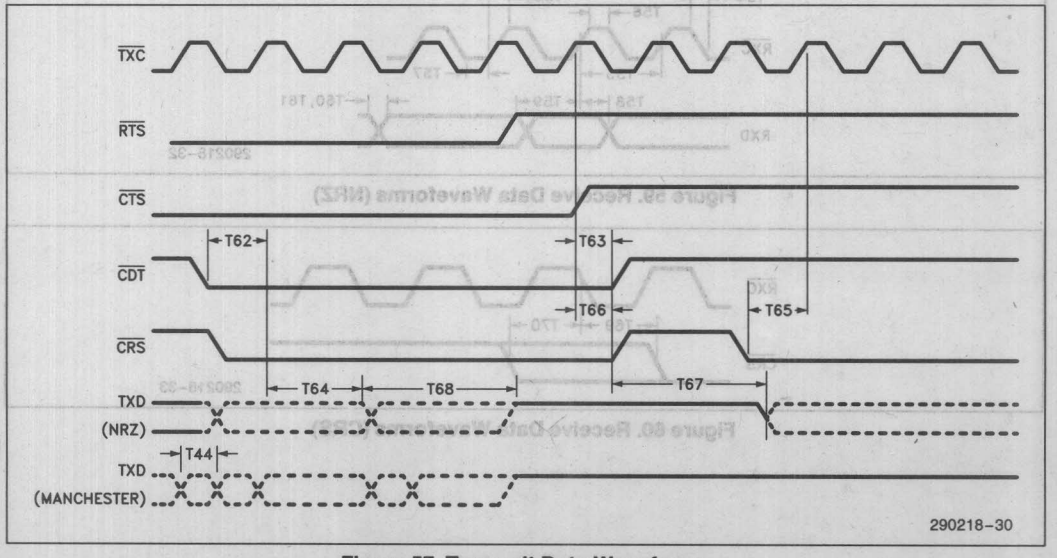


Figure 57. Transmit Data Waveforms

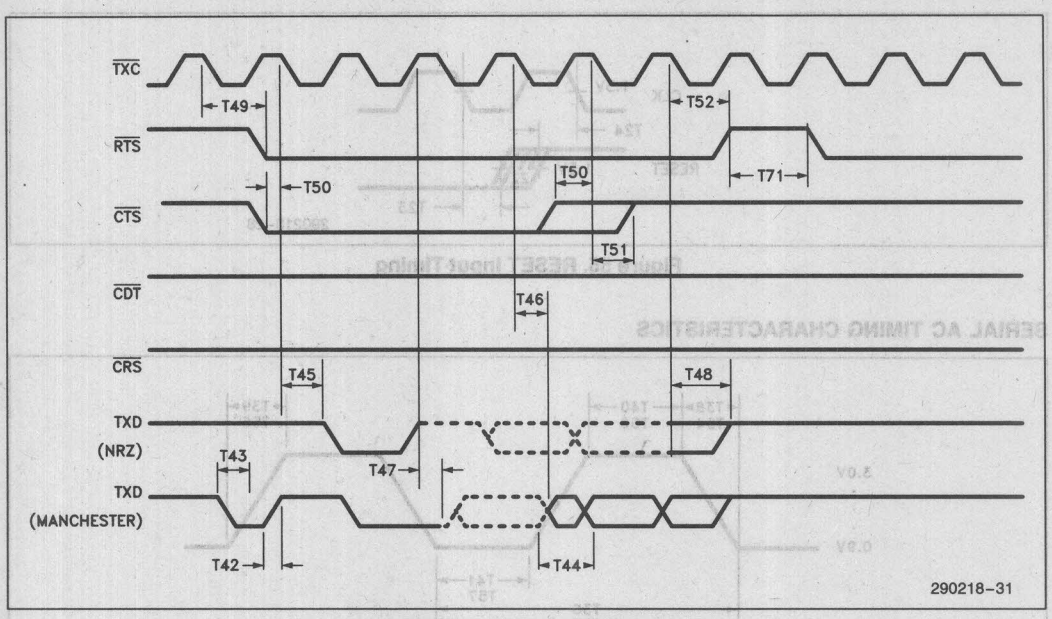


Figure 58. Transmit Data Waveforms

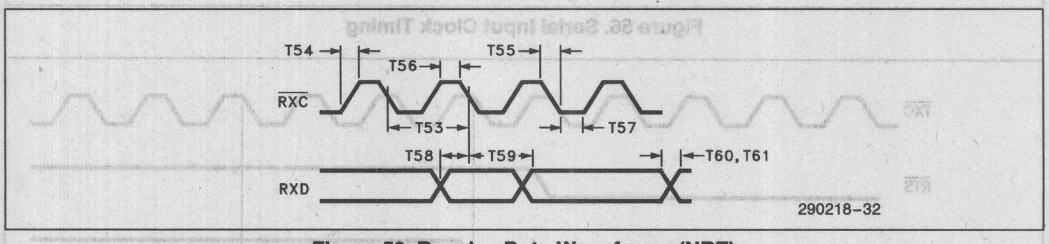


Figure 59. Receive Data Waveforms (NRZ)

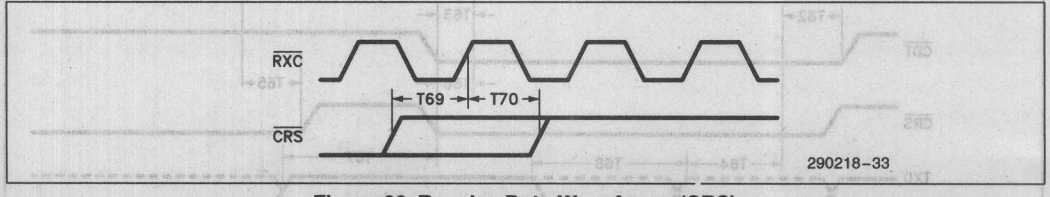
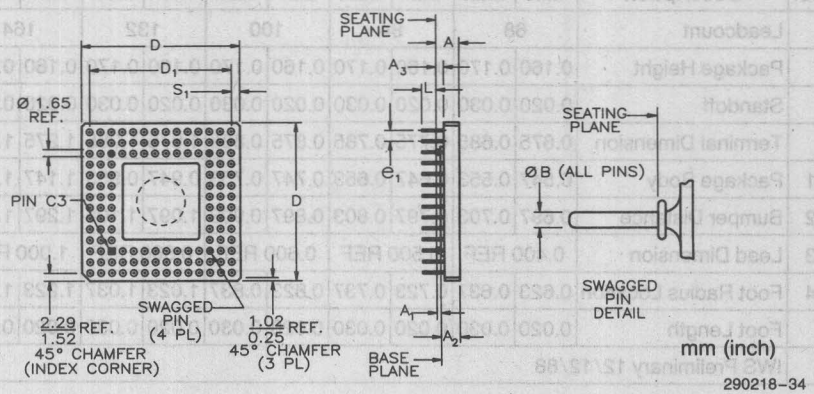


Figure 60. Receive Data Waveforms (CRS)

OUTLINE DIAGRAMS

132 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A



Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.76	1.27	Solid Lid	0.030	0.050	Solid Lid
A_2	2.67	3.43	Solid Lid	0.105	0.135	Solid Lid
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	36.45	37.21		1.435	1.465	
D_1	32.89	33.15		1.295	1.305	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N	132			132		
S_1	1.27	2.54		0.050	0.100	
ISSUE	IWS 10/12/88					

Intel Case Outline Drawings
Plastic Quad Flat Pack (PQFP)
0.025 Inch (0.635mm) Pitch

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170
A1	Standoff	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
D, E	Terminal Dimension	0.675	0.685	0.775	0.785	0.875	0.885	1.075	1.085	1.275	1.285	1.475	1.485
D1, E1	Package Body	0.547	0.553	0.647	0.653	0.747	0.753	0.947	0.953	1.147	1.153	1.347	1.353
D2, E2	Bumper Distance	0.697	0.703	0.797	0.803	0.897	0.903	1.097	1.103	1.297	1.303	1.497	1.503
D3, E3	Lead Dimension	0.400 REF		0.500 REF		0.600 REF		0.800 REF		1.000 REF		1.200 REF	
D4, E4	Foot Radius Location	0.623	0.637	0.723	0.737	0.823	0.837	1.023	1.037	1.223	1.237	1.423	1.437
L1	Foot Length	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
Issue	IWS Preliminary 12/12/88												INCH

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32
A1	Standoff	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
D, E	Terminal Dimension	17.15	17.40	19.69	19.94	22.23	22.48	27.31	27.56	32.39	32.64	37.47	37.72
D1, E1	Package Body	13.89	14.05	16.43	16.59	18.97	19.13	24.05	24.21	29.13	29.29	34.21	34.37
D2, E2	Bumper Distance	17.70	17.85	20.24	20.39	22.78	22.93	27.86	28.01	32.94	33.09	38.02	38.18
D3, E3	Lead Dimension	10.16 REF		12.70 REF		15.24 REF		20.32 REF		25.40 REF		30.48 REF	
D4, E4	Foot Radius Location	15.82	16.17	18.36	18.71	21.25	21.25	25.89	26.33	31.06	31.41	36.14	36.49
L1	Foot Length	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
Issue	IWS Preliminary 12/12/88												mm

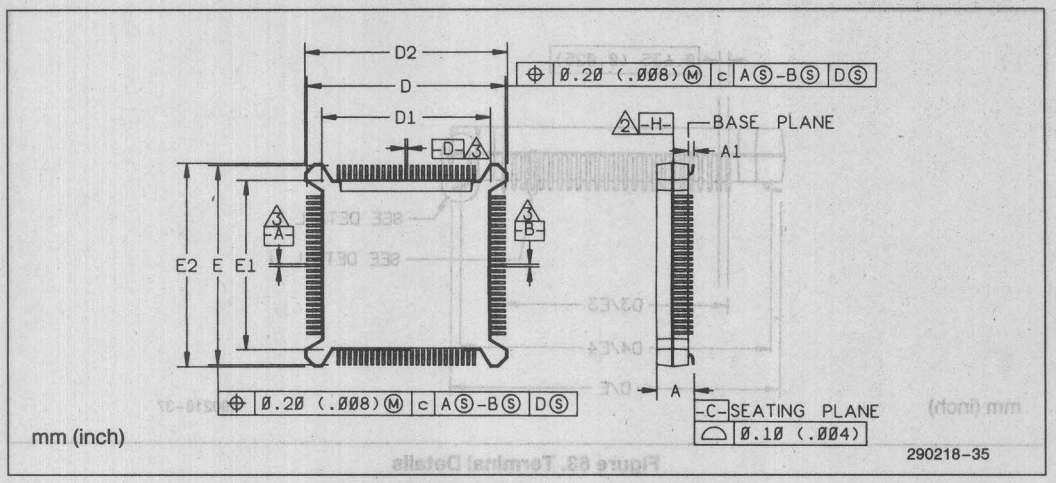


Figure 61. Principal Dimensions and Datums

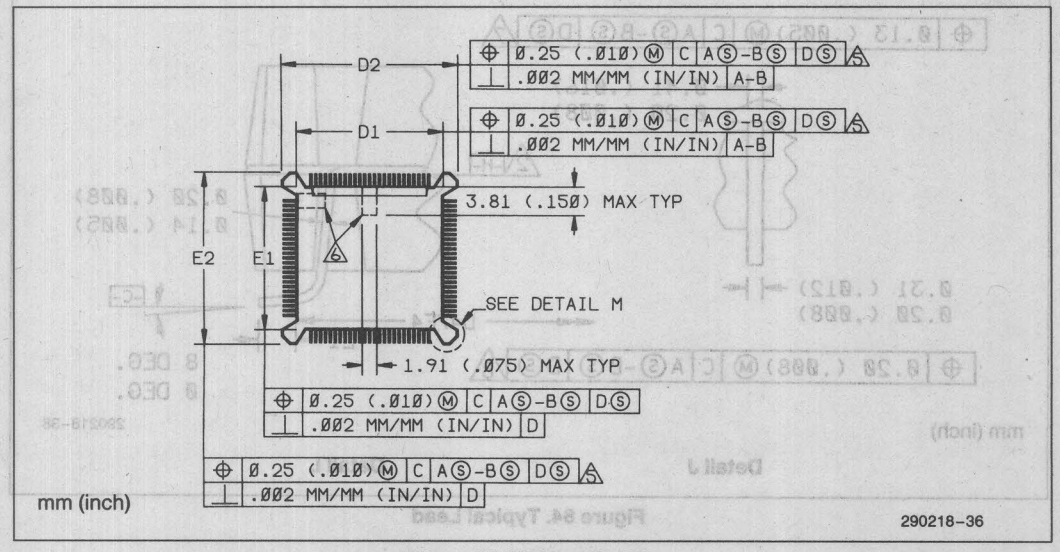


Figure 62. Molded Details

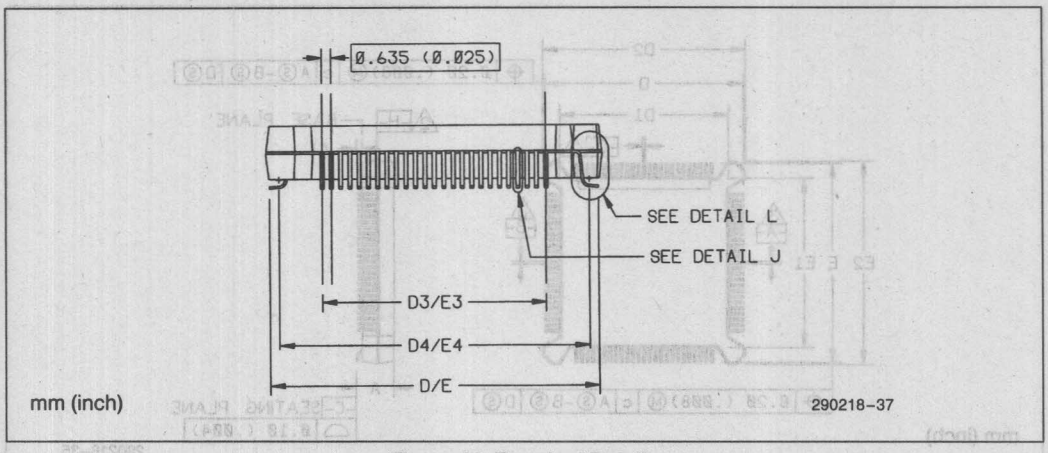


Figure 63. Terminal Details

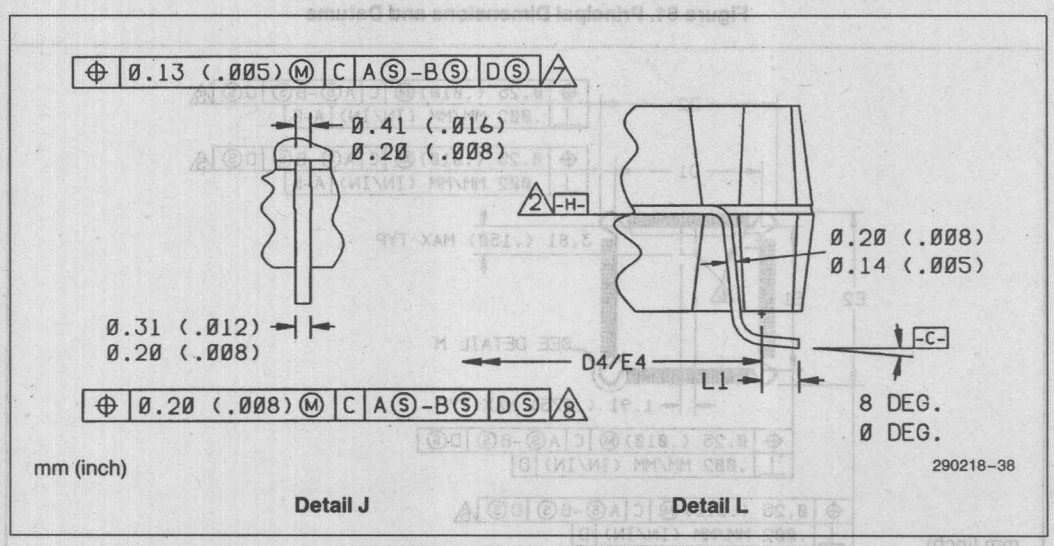


Figure 64. Typical Lead

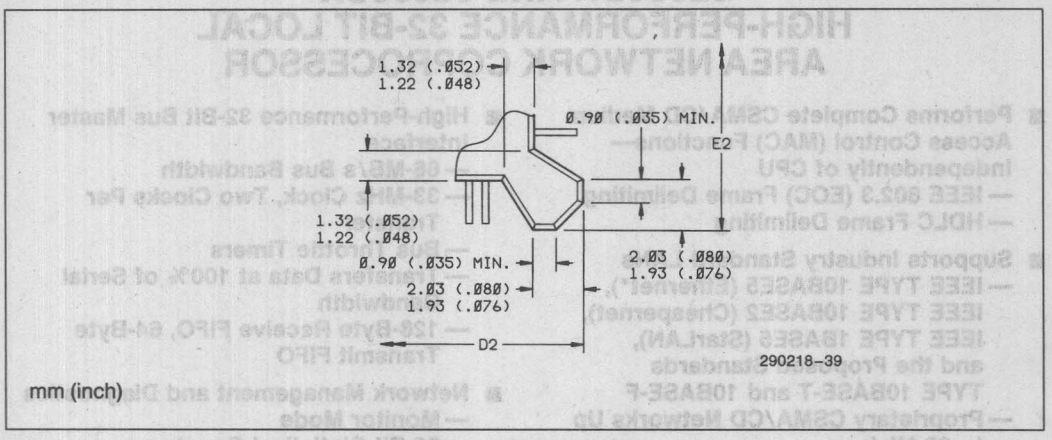


Figure 65. Detail M

REVISION HISTORY

The 82596 LAN Coprocessor data sheet version -003 contains updates and improvements to previous versions.

1. Added Pin Cross Reference table.

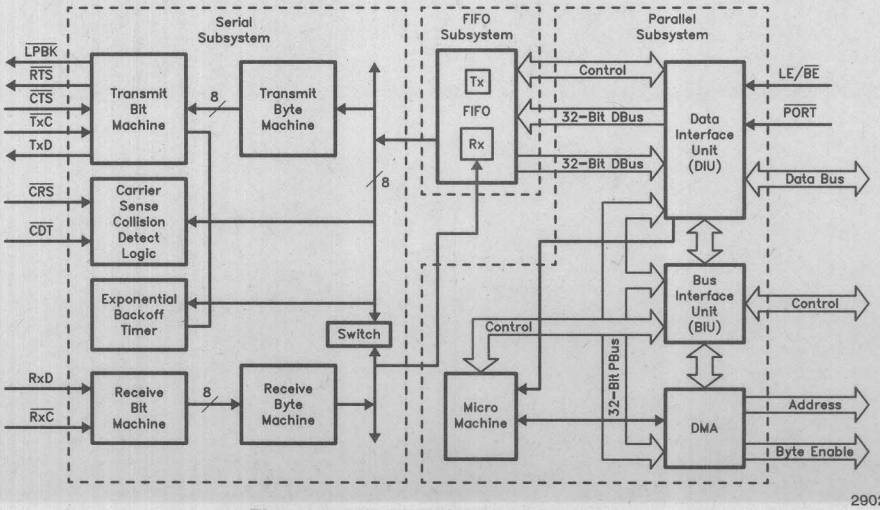
2. Added Bus Cycle figures.

82596DX AND 82596SX HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

- Performs Complete CSMA/CD Medium Access Control (MAC) Functions—Independently of CPU
  - IEEE 802.3 (EOC) Frame Delimiting
  - HDLC Frame Delimiting
- Supports Industry Standard LANs
  - IEEE TYPE 10BASE5 (Ethernet*), IEEE TYPE 10BASE2 (Cheapernet), IEEE TYPE 1BASE5 (StarLAN), and the Proposed Standards TYPE 10BASE-T and 10BASE-F
  - Proprietary CSMA/CD Networks Up to 20 Mb/s
- On-Chip Memory Management
  - Automatic Buffer Chaining
  - Buffer Reclamation after Receipt of Bad Frames; Optional Save Bad Frames
  - 32-Bit Segmented or Linear (Flat) Memory Addressing Formats
- 82586 Software Compatible
- Optimized CPU Interface
  - Optimized Bus Interface to Intel's 32-Bit 386™DX and 16-Bit 386™SX and 376™ Microprocessors
  - Supports Big Endian and Little Endian Byte Ordering
- High-Performance 32-Bit Bus Master Interface
  - 66-MB/s Bus Bandwidth
  - 33-MHz Clock, Two Clocks Per Transfer
  - Bus Throttle Timers
  - Transfers Data at 100% of Serial Bandwidth
  - 128-Byte Receive FIFO, 64-Byte Transmit FIFO
- Network Management and Diagnostics
  - Monitor Mode
  - 32-Bit Statistical Counters
- Self-Test Diagnostics
- Configurable Initialization Root for Data Structures
- High-Speed, 5-V, CHMOS** IV Technology
- 132-Pin Plastic Quad Flat Pack (PQFP) and PGA Package

  (See Packaging Spe. Order No. 231369)

386™ is a trademark of Intel Corporation
*Ethernet is a registered trademark of Xerox Corporation.
**CHMOS is a patented process of Intel Corporation.



290219-1

Figure 1. 82596DX/SX Block Diagram

October 1990
Order Number: 290219-003

82596DX AND 82596SX HIGH-PERFORMANCE 32-BIT LOCAL AREA NETWORK COPROCESSOR

CONTENTS

PAGE

INTRODUCTION 1-112

PIN DESCRIPTIONS 1-119

82596 AND HOST CPU INTERACTION 1-123

82596 BUS INTERFACE 1-123

82596 MEMORY ADDRESSING 1-123

82596 SYSTEM MEMORY STRUCTURE 1-125

TRANSMIT AND RECEIVE MEMORY STRUCTURES 1-126

TRANSMITTING FRAMES 1-129

RECEIVING FRAMES 1-130

82596 NETWORK MANAGEMENT AND DIAGNOSTICS 1-130

NETWORK PLANNING AND MAINTENANCE 1-132

STATION DIAGNOSTICS AND SELF-TEST 1-133

82586 SOFTWARE COMPATIBILITY 1-133

INITIALIZING THE 82596 1-133

SYSTEM CONFIGURATION POINTER (SCP) 1-133

Writing the Sysbus 1-134

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP) 1-135

INITIALIZATION PROCESS 1-135

CONTROLLING THE 82596DX/SX 1-136

82596 CPU ACCESS INTERFACE (PORT#) 1-136

MEMORY ADDRESSING FORMATS 1-137

LITTLE ENDIAN AND BIG ENDIAN BYTE ORDERING 1-138

COMMAND UNIT (CU) 1-138

RECEIVE UNIT (RU) 1-138

CONTENTS

PAGE

SYSTEM CONTROL BLOCK (SCB) 1-138

SCB OFFSET ADDRESS 1-141

CBL Offset (Address) 1-141

RFA Offset (Address) 1-142

SCB STATISTICAL COUNTERS 1-142

Statistical Counter Operation 1-142

ACTION COMMANDS AND OPERATING MODES 1-143

NOP 1-143

Individual Address Setup 1-144

Configure 1-145

Multicast-Setup 1-151

Transmit 1-152

Jamming Rules 1-154

TDR 1-155

Dump 1-157

Diagnose 1-160

RECEIVE FRAME DESCRIPTOR 1-160

Simplified Memory Structure 1-161

Flexible Memory Structure 1-162

Receive Buffer Descriptor (RBD) 1-163

ELECTRICAL AND TIMING CHARACTERISTICS 1-168

DC Characteristics 1-168

AC Characteristics 1-169

82596DX Input/Output System Timings 1-169

82596SX Input/Output System Timings 1-171

Transmit/Receive Clock Parameters 1-173

82596DX/SX BUS OPERATION 1-175

System Interface A.C. Timing Characteristics 1-176

Input Waveforms 1-177

Serial A.C. Timing Characteristics 1-179

OUTLINE DIAGRAMS 1-181

REVISION HISTORY 1-185

1

INTRODUCTION

The 82596DX/SX is an intelligent, high-performance 32-bit Local Area Network coprocessor. The 82596DX/SX implements the CSMA/CD access method and can be configured to support all existing IEEE 802.3 standards—TYPEs 10BASE5, 10BASE2, 1BASE5, and 10BROAD36. It can also be used to implement the proposed standards TYPE 10BASE-T and 10BASE-F. The 82596DX/SX performs high-level commands, command chaining, and interprocessor communications via shared memory, thus relieving the host CPU of many tasks associated with network control. All time-critical functions are performed independently of the CPU, this increases network performance and efficiency. The 82596DX/SX bus interface is optimized for Intel's 386™ DX, 386 SX and 376™ microprocessors.

The 82596DX/SX implements all IEEE 802.3 Medium Access Control and channel interface functions, these include framing, preamble generation and stripping, source address generation, destination address checking, short-frame detection, and automatic length-field handling. Data rates up to 20 Mb/s are supported.

The 82596DX/SX provides a powerful host system interface. It manages memory structures automatically, with command chaining and bidirectional data chaining. An on-chip DMA controller manages four channels, this allows autonomous transfer of data blocks (buffers and frames) and relieves the CPU of byte transfer overhead. Buffers containing errored or collided frames can be automatically recovered without CPU intervention. The 82596DX/SX provides an upgrade path for existing 82586 software drivers by providing an 82586-software-compatible mode that supports the current 82586 memory structure. The 82596DX/SX also has a Flexible memory structure and a Simplified memory structure. The 82596DX/SX can address up to 4 gigabytes of memory. The 82596DX/SX supports Little Endian and Big Endian byte ordering.

The 82596DX/SX bus interface is optimized to Intel's 386 DX, 386 SX, and 376 microprocessors, providing a bus transfer rate of up to 66 MB/s at 33 MHz. The bus interface employs bus throttle timers to regulate 82596DX/SX bus use. Two large, independent FIFOs—128 bytes for Receive and 64 bytes for Transmit—tolerate long bus latencies and provide programmable thresholds that allow the user to optimize bus overhead for any worst-case bus latency.

The 82596DX/SX provides a wide range of diagnostics and network management functions, these include internal and external loopback, exception condition tallies, channel activity indicators, optional capture of all frames regardless of destination address (promiscuous mode), optional capture of errored or collided frames, and time domain reflectometry for locating fault points on the network cable. The statistical counters, in 32-bit segmented and linear modes, are 32-bits each and include CRC errors, alignment errors, overrun errors, resource errors, short frames, and received collisions. The 82596DX/SX also features a monitor mode for network analysis. In this mode the 82596DX/SX can capture status bytes, and update statistical counters, of frames monitored on the link without transferring the contents of the frames to memory. This can be done concurrently while transmitting and receiving frames destined for that station.

The 82596DX/SX can be used in both baseband and broadband networks. It can be configured for maximum network efficiency (minimum contention overhead) with networks of any length. Its highly flexible CSMA/CD unit supports address field lengths of zero through six bytes—configurable to either IEEE 802.3/Ethernet or HDLC frame delimitation. It also supports 16- or 32-bit cyclic redundancy checks. The CRC can be transferred directly to memory for receive, operations or dynamically inserted for transmit operations. The CSMA/CD unit can also be configured for full duplex operation for high throughput in point-to-point connections.

The 82596DX/SX is fabricated with Intel's reliable, 5-V, CHMOS IV technology. It is available in a 132-pin PQFP or PGA package.

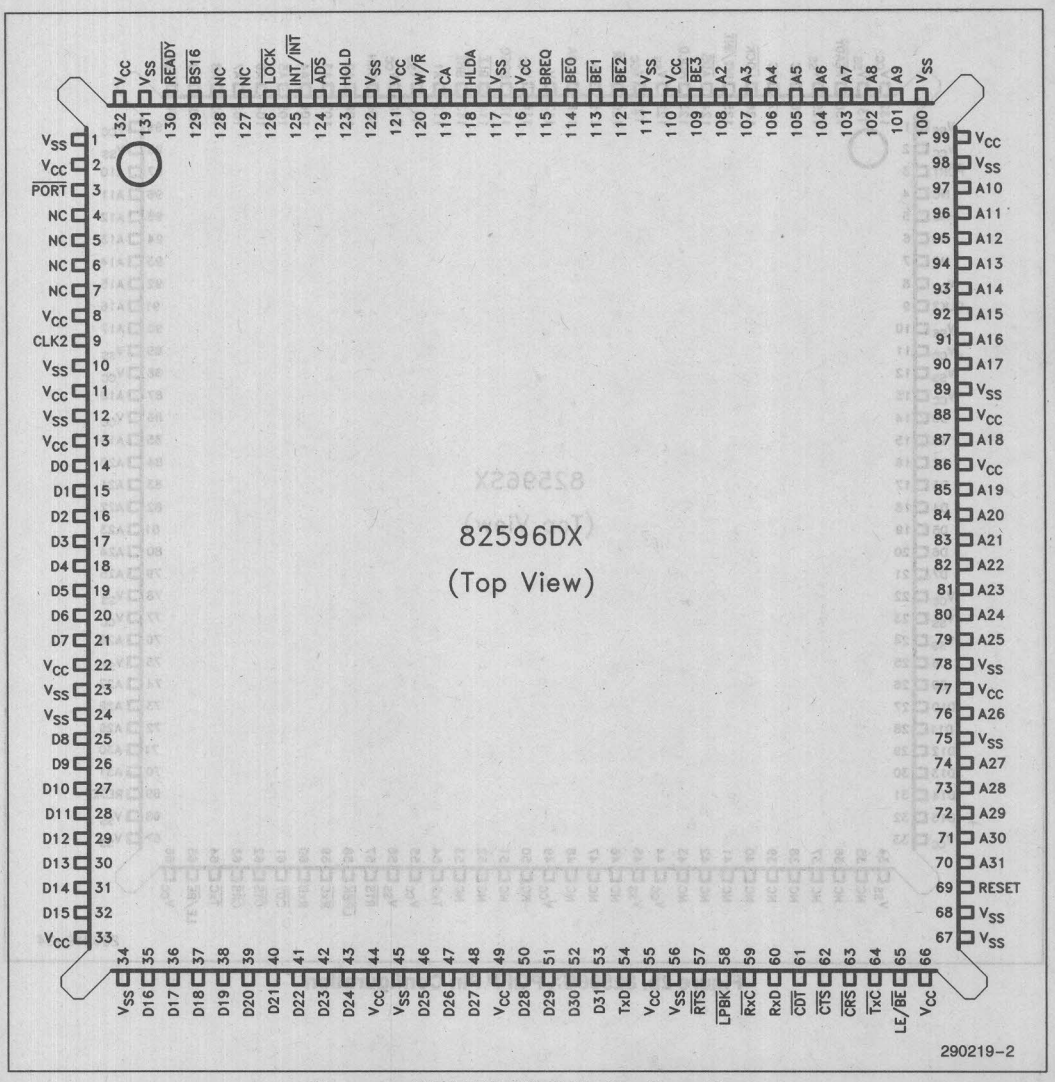


Figure 2a. 82596DX PQFP Pin Configuration

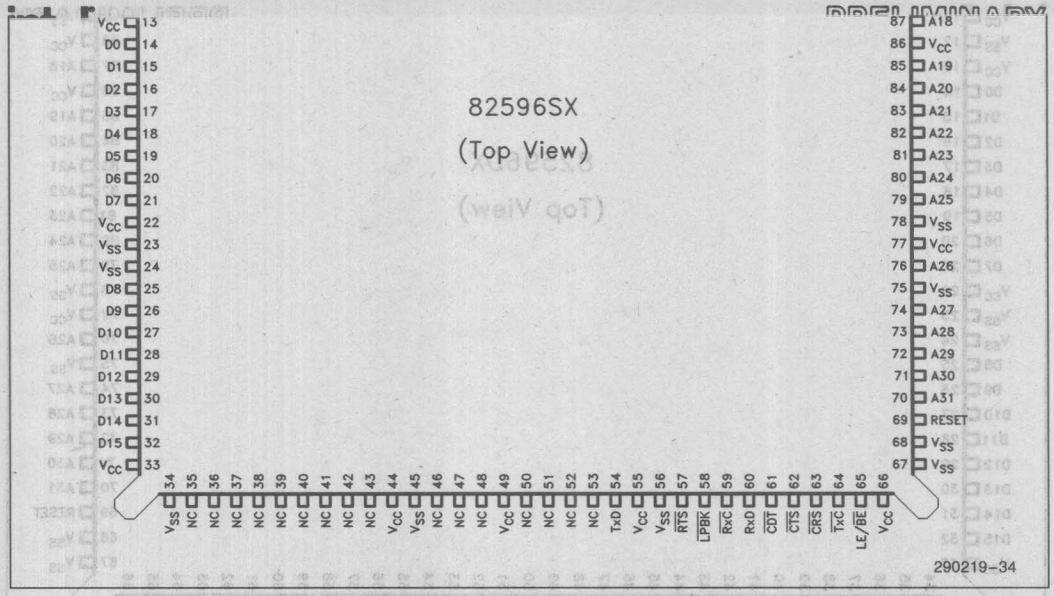


Figure 2b. 82596SX PQFP Pin Configuration

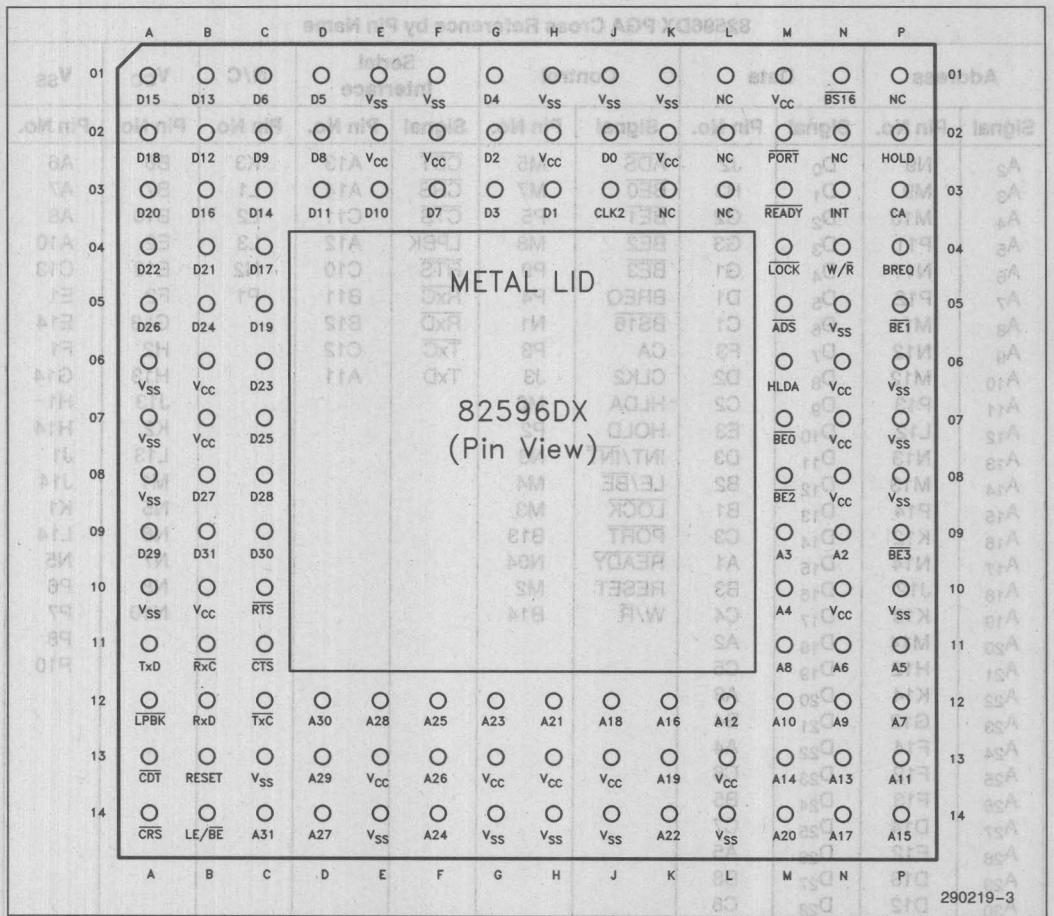


Figure 3a. 82596DX PGA Pin View Side

82596DX PGA Cross Reference by Pin Name

Address		Data		Control		Serial Interface		N/C	V_{CC}	V_{SS}
Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Pin No.	Pin No.	Pin No.
A_2	N9	D_0	J2	$\overline{ADS}$	M5	$\overline{CDT}$	A13	K3	B6	A6
A_3	M9	D_1	H3	$\overline{BE0}$	M7	$\overline{CRS}$	A14	L1	B7	A7
A_4	M10	D_2	G2	$\overline{BE1}$	P5	$\overline{CTS}$	C11	L2	B10	A8
A_5	P11	D_3	G3	$\overline{BE2}$	M8	$\overline{LPBK}$	A12	L3	E2	A10
A_6	N11	D_4	G1	$\overline{BE3}$	P9	$\overline{RTS}$	C10	N2	E13	C13
A_7	P12	D_5	D1	BREQ	P4	$\overline{RxC}$	B11	P1	F2	E1
A_8	M11	D_6	C1	$\overline{BS16}$	N1	$\overline{RxD}$	B12		G13	E14
A_9	N12	D_7	F3	CA	P3	$\overline{TxC}$	C12		H2	F1
A_{10}	M12	D_8	D2	CLK2	J3	TxD	A11		H13	G14
A_{11}	P13	D_9	C2	HLDA	M6				J13	H1
A_{12}	L12	D_{10}	E3	HOLD	P2				K2	H14
A_{13}	N13	D_{11}	D3	INT/$\overline{INT}$	N3				L13	J1
A_{14}	M13	D_{12}	B2	LE/$\overline{BE}$	M4				M1	J14
A_{15}	P14	D_{13}	B1	$\overline{LOCK}$	M3				N5	K1
A_{16}	K12	D_{14}	C3	$\overline{PORT}$	B13				N6	L14
A_{17}	N14	D_{15}	A1	$\overline{READY}$	N04				N7	N5
A_{18}	J12	D_{16}	B3	RESET	M2				N8	P6
A_{19}	K13	D_{17}	C4	W/$\overline{R}$	B14				N10	P7
A_{20}	M14	D_{18}	A2							P8
A_{21}	H12	D_{19}	C5							P10
A_{22}	K14	D_{20}	A3							
A_{23}	G12	D_{21}	B4							
A_{24}	F14	D_{22}	A4							
A_{25}	F12	D_{23}	C6							
A_{26}	F13	D_{24}	B5							
A_{27}	D14	D_{25}	C7							
A_{28}	E12	D_{26}	A5							
A_{29}	D13	D_{27}	B8							
A_{30}	D12	D_{28}	C8							
A_{31}	C14	D_{29}	A9							
		D_{30}	C9							
		D_{31}	B9							

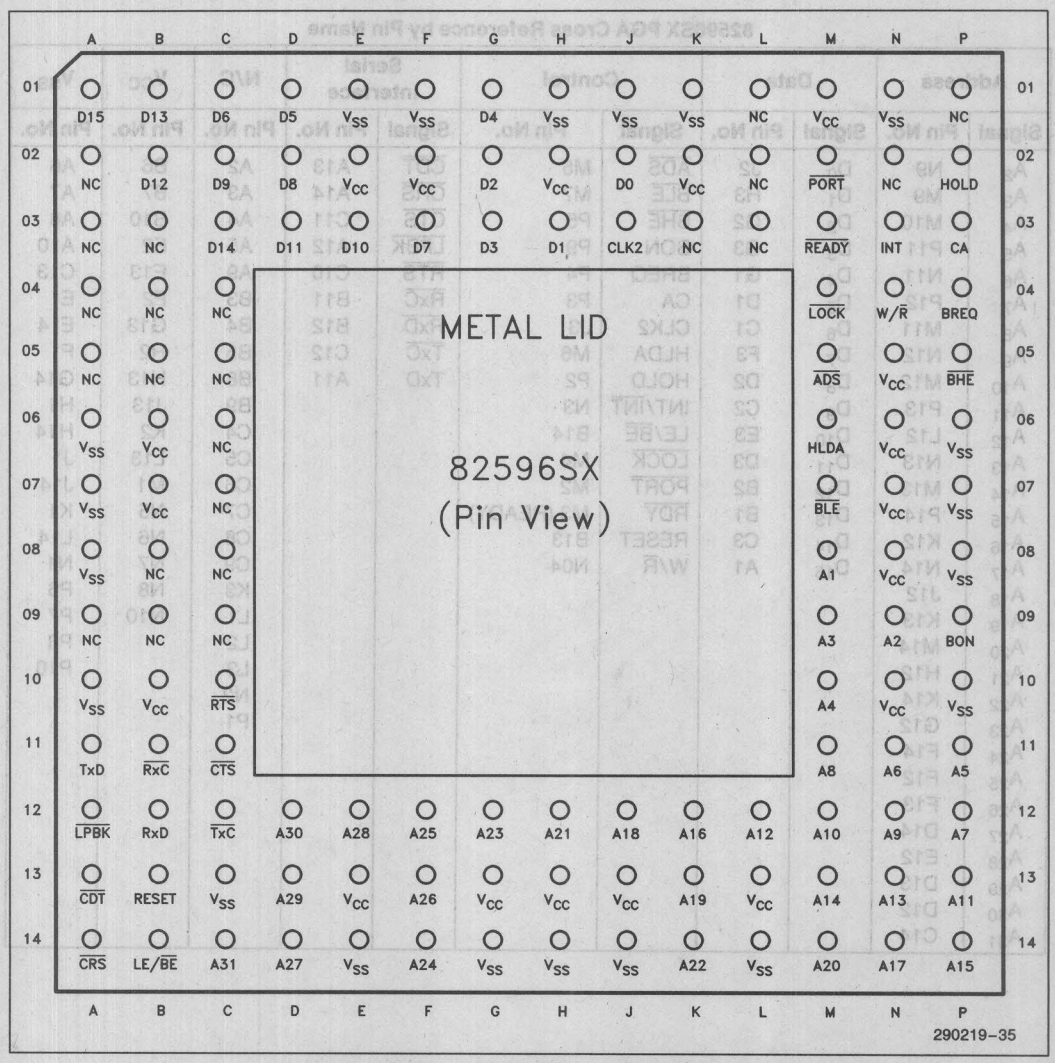


Figure 3b. 82596SX PGA Pin View Side

82596SX PGA Cross Reference by Pin Name

Address		Data		Control		Serial Interface		N/C	V_{CC}	V_{SS}
Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Pin No.	Pin No.	Pin No.
A_2	N9	D_0	J2	$\overline{ADS}$	M5	$\overline{CDT}$	A13	A2	B6	A6
A_3	M9	D_1	H3	$\overline{BLE}$	M7	$\overline{CRS}$	A14	A3	B7	A7
A_4	M10	D_2	G2	$\overline{BHE}$	P5	$\overline{CTS}$	C11	A4	B10	A8
A_5	P11	D_3	G3	BON	P9	$\overline{LPBK}$	A12	A5	E2	A10
A_6	N11	D_4	G1	BREQ	P4	$\overline{RTS}$	C10	A9	E13	C13
A_7	P12	D_5	D1	CA	P3	$\overline{RxC}$	B11	B3	F2	E1
A_8	M11	D_6	C1	CLK2	J3	$\overline{RxD}$	B12	B4	G13	E14
A_9	N12	D_7	F3	HLDA	M6	$\overline{TxC}$	C12	B5	H2	F1
A_{10}	M12	D_8	D2	HOLD	P2	TxD	A11	B8	H13	G14
A_{11}	P13	D_9	C2	INT/$\overline{INT}$	N3			B9	J13	H1
A_{12}	L12	D_{10}	E3	LE/$\overline{BE}$	B14			C4	K2	H14
A_{13}	N13	D_{11}	D3	$\overline{LOCK}$	M4			C5	L13	J1
A_{14}	M13	D_{12}	B2	$\overline{PORT}$	M2			C6	M1	J14
A_{15}	P14	D_{13}	B1	$\overline{RDY}$	M3 (READY)			C7	N5	K1
A_{16}	K12	D_{14}	C3	RESET	B13			C8	N6	L14
A_{17}	N14	D_{15}	A1	W/$\overline{R}$	N04			C9	N7	N1
A_{18}	J12							K3	N8	P6
A_{19}	K13							L1	N10	P7
A_{20}	M14							L2		P8
A_{21}	H12							L3		P10
A_{22}	K14							N2		
A_{23}	G12							P1		
A_{24}	F14									
A_{25}	F12									
A_{26}	F13									
A_{27}	D14									
A_{28}	E12									
A_{29}	D13									
A_{30}	D12									
A_{31}	C14									

			These lines are inputs during a CPU Port access; in this mode the CPU writes the next address to the 82596 through the Data lines. During $\overline{PORT}$ commands (Relocatable SCP, Self-Test, and Dump) the address must be aligned to a 16 byte boundary. This frees the D_3–D_0 lines so they can be used to distinguish the commands. The following is a summary of the decoding data.

D0	D1	D2	D3	D4–D31	Function
0	0	0	0	0000	Reset
0	1	0	0	ADDR	Relocatable SCP
1	0	0	0	ADDR	Self-Test
1	1	0	0	ADDR	Dump Command

(D15–D0)			These 16 Data Bus lines are bidirectional, tri-state lines that provide the entire data path for the 82596SX. In the 82596SX D16–D31 are not connected (NC).
A31–A2	70–108	O	**ADDRESS LINES.** These 30 tri-stated Address lines output the address bits required for memory operation. These lines are floated after a Reset or when the bus is not acquired.
A1	112	O	The 82596SX requires this additional address line to output the address bits required for memory operation.
$\overline{BE3}$–$\overline{BE0}$	109–114	O	**BYTE ENABLE.** (82596DX only.) These tri-stated signals are used to indicate which bytes are involved with the current memory access. The number of Byte Enable signals asserted indicates the physical size of the data being transferred (1, 2, 3, or 4 bytes). • $\overline{BE0}$ indicates D0–D7 • $\overline{BE1}$ indicates D8–D15 • $\overline{BE2}$ indicates D16–D23 • $\overline{BE3}$ indicates D24–D31 These lines are floated after a Reset or when the bus is not acquired.
$\overline{BHE}$, $\overline{BLE}$	109–114	O	(82596SX only.) These signals are the Byte High Enable and Byte Low Enable signals for the 82596SX.
BON	109	O	**BUS ON.** (82596SX only.) This signal is driven high when the 82596 is holding the bus. This signal is tri-stated when the bus is relinquished. BON has the same timing as the Byte Enables.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
W/$\overline{R}$	120	O	**WRITE/READ.** This dual-function pin is used to distinguish Write and Read cycles. This line is floated after a Reset or when the bus is not acquired.
$\overline{ADS}$	124	O	**ADDRESS STATUS.** This tri-state pin is used by the 82596 to indicate that a valid bus cycle has begun and that A31–A2, $\overline{BE3}$–$\overline{BE0}$, and W/$\overline{R}$ are being driven. It is asserted during t1 bus states. This line is floated after a Reset or when the bus is not acquired.
$\overline{RDY}$	130	I	**READY.** Active low. This signal is the acknowledgment from addressed memory that the transfer cycle can be completed. When high, it causes wait states to be inserted. It is ignored at the end of the first clock of the bus cycle's data cycle. This active-low signal does not have an internal pull-up resistor. This signal must meet the setup and hold times to operate correctly.
$\overline{LOCK}$	126	O	**LOCK.** This tri-state pin is used to distinguish locked and unlocked bus cycles. $\overline{LOCK}$ generates a semaphore handshake to the CPU. $\overline{LOCK}$ can be active for several memory cycles, it goes active during the first locked memory cycle (t1) and goes inactive at the last locked cycle (t2). This line is floated after a Reset or when the bus is not acquired. $\overline{LOCK}$ can be disabled via the sysbus byte in software.
$\overline{BS16}$	129	I	**BUS SIZE.** This signal allows the 82596DX to work with either 16- or 32-bit bytes. This signal is static and should be tied high for 32-bit operation or low for 16-bit operation. In Little Endian mode the D0–D15 lines are driven when $\overline{BS16}$ is inserted, in Big Endian mode the D16–D31 lines are driven.
HOLD	123	O	**HOLD.** The HOLD signal is active high, the 82596 uses it to request local bus mastership. In normal operation HOLD goes inactive before HLDA. The 82596 can be forced off the bus by deasserting HLDA or if the bus throttle timers expire.
HLDA	118	I	**HOLD ACKNOWLEDGE.** The HLDA signal is active high, it indicates that bus mastership has been given to the 82596. HLDA is internally synchronized; after HOLD is detected low, the CPU drives HLDA low. **NOTE** *Do not connect HLDA to V_{CC}—it will cause a deadlock.* A user wanting to give the 82596 permanent access to the bus should connect HLDA to HOLD. If HLDA goes inactive before HOLD, the 82596 will release the bus (by deasserting HOLD) within a specified number of system clocks.
BREQ	115	I	**BUS REQUEST.** This signal, when configured to an externally activated mode, is used to trigger the bus throttle timers.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{PORT}$	3	I	**PORT.** When this signal is received, the 82596 latches the data on the data bus into an internal 32-bit register. When the CPU is asserting this signal it can write into the 82596 (via the data bus). This pin must be activated twice during all CPU Port access commands.
RESET	69	I	**RESET.** This active high, internally synchronized signal causes the 82596 to terminate current activity. The signal must be high for at least five system clock cycles. After five system clock cycles and four TxC clock cycles the 82596 will execute a Reset when it receives a high RESET signal. When RESET returns to low, the 82596 waits for the first CA signal and then begins the initialization sequence.
LE/$\overline{BE}$	65	I	**LITTLE ENDIAN/BIG ENDIAN.** This dual-function pin is used to select byte ordering. When LE/$\overline{BE}$ is high, little endian byte ordering is used; when low, big endian byte ordering is used for data in frames (bytes) and for control (SCB, RFD, CBL, etc.).
CA	119	I	**CHANNEL ATTENTION.** The CPU uses this pin to force the 82596 to begin executing memory resident Command blocks. The CA signal is internally synchronized. The signal must be high for at least one system clock. It is latched internally on the high to low edge and then detected by the 82596. The first CA after a Reset forces the 82596 into the initialization sequence beginning at location 00FFFFF6h or an SCP address written to the 82596 using CPU Port access. All subsequent CA signals cause the 82596 to begin executing new command sequences from the SCB.
INT/$\overline{INT}$	125	O	**INTERRUPT.** A high signal on this pin notifies the CPU that the 82596 is requesting an interrupt. This signal is an edge triggered interrupt signal, and can be configured to be active high or low.
V_{CC}	18 Pins		**POWER.** +5V ±10%.
V_{SS}	18 Pins		**GROUND.** 0V.
TxD	54	O	**TRANSMIT DATA.** This pin transmits data to the serial link. It is high when not transmitting.
$\overline{TxC}$	64	I	**TRANSMIT CLOCK.** This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the high to low clock transition. For Manchester encoding, the transmitted bit center is aligned with the low to high transition. Transmit clock should always be running for proper device operation.

PIN DESCRIPTIONS (Continued)

Symbol	PQFP Pin No.	Type	Name and Function
$\overline{\text{LPBK}}$	58	O	**LOOPBACK.** This TTL-level control signal enables the loopback mode. In this mode serial data on the TxD input is routed through the 82C501 internal circuits and back to the RxD output without driving the transceiver cable. To enable this signal, both internal and external loopback need to be set with the Configure command.
RxD	60	I	**RECEIVE DATA.** This pin receives NRZ serial data only. It must be high when not receiving.
$\overline{\text{RxC}}$	59	I	**RECEIVE CLOCK.** This signal provides timing information to the internal shifting logic. For NRZ data the state of the RxD pin is sampled on the high to low transition of the clock.
$\overline{\text{RTS}}$	57	O	**REQUEST TO SEND.** When this signal is low the 82596 informs the external interface that it has data to transmit. It is forced high after a Reset or when transmission is stopped.
$\overline{\text{CTS}}$	62	I	**CLEAR TO SEND.** An active-low signal that enables the 82596 to send data. It is normally used as an interface handshake to $\overline{\text{RTS}}$. Asserting $\overline{\text{CTS}}$ high stops transmission. $\overline{\text{CTS}}$ is internally synchronized. If $\overline{\text{CTS}}$ goes inactive, meeting the setup time to the $\overline{\text{TxC}}$ negative edge, the transmission will stop and $\overline{\text{RTS}}$ will go inactive within, at most, two $\overline{\text{TxC}}$ cycles.
$\overline{\text{CRS}}$	63	I	**CARRIER SENSE.** This signal is active low, it is used to notify the 82596 that traffic is on the serial link. It is only used if the 82596 is configured for external Carrier Sense. In this configuration external circuitry is required for detecting traffic on the serial link. $\overline{\text{CRS}}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CRSF = 0).
$\overline{\text{CDT}}$	61	I	**COLLISION DETECT.** This active-low signal informs the 82596 that a collision has occurred. It is only used if the 82596 is configured for external Collision Detect. External circuitry is required for collision detection. $\overline{\text{CDT}}$ is internally synchronized. To be accepted, the signal must remain active for at least two serial clock cycles (for CDTF = 0).

82596 AND HOST CPU INTERACTION

The 82596DX/SX and the host CPU communicate through shared memory. Because of its on-chip DMA capability, the 82596 can make data block transfers (buffers and frames) independently of the CPU; this greatly reduces the CPU byte transfer overhead.

NOTE:
The 82596DX and 82596SX differ in their address pin definitions and their data bus sizes. Information in this data sheet applies to both versions unless otherwise stated.

The 82596 is a multitasking coprocessor that comprises two independent logical units—the Command Unit (CU) and the Receive Unit (RU). The CU executes commands from shared memory. The RU handles all activities related to frame reception. The independence of the CU and RU enables the 82596 to engage in both activities simultaneously—the CU can fetch and execute commands from memory while the RU is storing received frames in memory. The CPU is only involved with this process after the CU has executed a sequence of commands or the RU has finished storing a sequence of frames.

The CPU and the 82596 use the hardware signals Interrupt (INT) and Channel Attention (CA) to initiate communication with the System Control Block (SCB), see Figure 4. The 82596 uses INT to alert the CPU of a change in the contents of the SCB, the CPU uses CA to alert the 82596.

The 82596 has a CPU Port Access state that allows the CPU to execute certain functions without accessing memory. The 82596 PORT pin and data bus pins are used to enable this feature. The CPU can directly activate four operations when the 82596 is in this state.

- Write an alternative System Configuration Pointer (SCP). This can be used when the 82596 cannot use the default SCP address space.
- Write a different Dump Command Pointer and execute Dump. This can be used for troubleshooting No Response problems.
- The CPU can reset the 82596 via software without disturbing the rest of the system.
- A self-test can be used for board testing; the 82596 will execute a self-test and write the results to memory.

82596 BUS INTERFACE

The 82596DX/SX has bus interface timings and pin definitions that are compatible with Intel's 32-bit 386 DX, 386 SX, and 376 microprocessors. This eliminates the need for additional bus interface logic. Operating at 33 MHz, the 82596's bus bandwidth can be as high as 66 MB/s. Since Ethernet only requires 1.25 MB/s, this leaves a considerable amount of bandwidth for the CPU. The 82596 also has a bus throttle to regulate its use of the bus. Two timers can be programmed through the SCB: one controls the maximum time the 82596 can remain on the bus, the other controls the time the 82596 must stay off the bus (see Figure 5). The bus throttle can be programmed to trigger internally with HLDA or externally with BREQ. These timers can restrict the 82596 HOLD activation time and improve bus utilization.

82596 MEMORY ADDRESSING

The 82596 has a 32-bit memory address range, which allows addressing up to four gigabytes of memory. The 82596 has three memory addressing modes (see Table 1).

- **82586 Mode.** The 82596 has a 24-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-kB memory segment. Transmit and Receive buffers can reside in a 24-bit address space.
- **32-Bit Segmented Mode.** The 82596 has a 32-bit memory address range. The System Control Block, Command List, Receive Descriptor List, and Buffer Descriptors must reside in one 64-kB memory segment. Transmit and Receive buffers can reside in a 32-bit address space.
- **Linear Mode.** The 82596 has a 32-bit memory address range. Any memory structure can reside anywhere within the 32-bit memory address range.

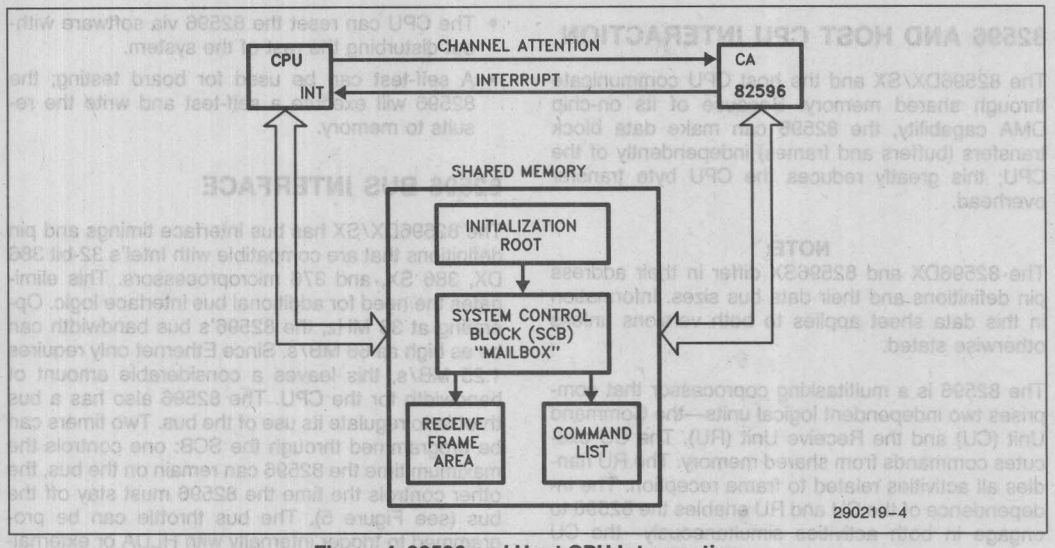


Figure 4. 82596 and Host CPU Intervention

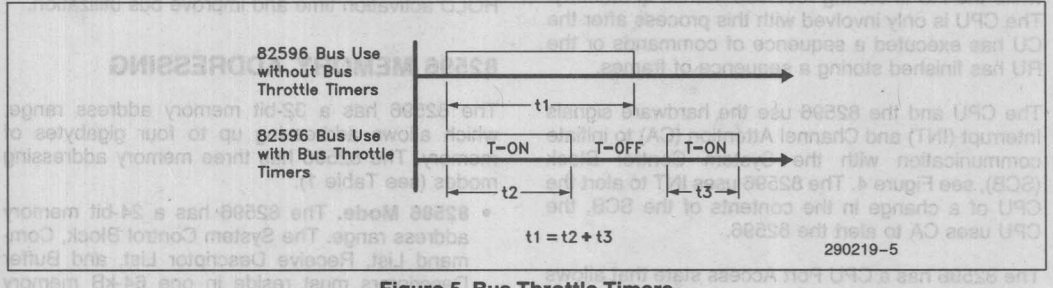


Figure 5. Bus Throttle Timers

Table 1. 82596 Memory Addressing Formats

Pointer or Offset	Operation Mode		
	82586	32-Bit Segmented	Linear
ISCP ADDRESS	24-Bit Linear	32-Bit Linear	32-Bit Linear
SCB ADDRESS	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Command Block Pointers	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Frame Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Tx Buffer Descriptors	Base (24) + Offset (16)	Base (32) + Offset (16)	32-Bit Linear
Rx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear
Tx Buffers	24-Bit Linear	32-Bit Linear	32-Bit Linear

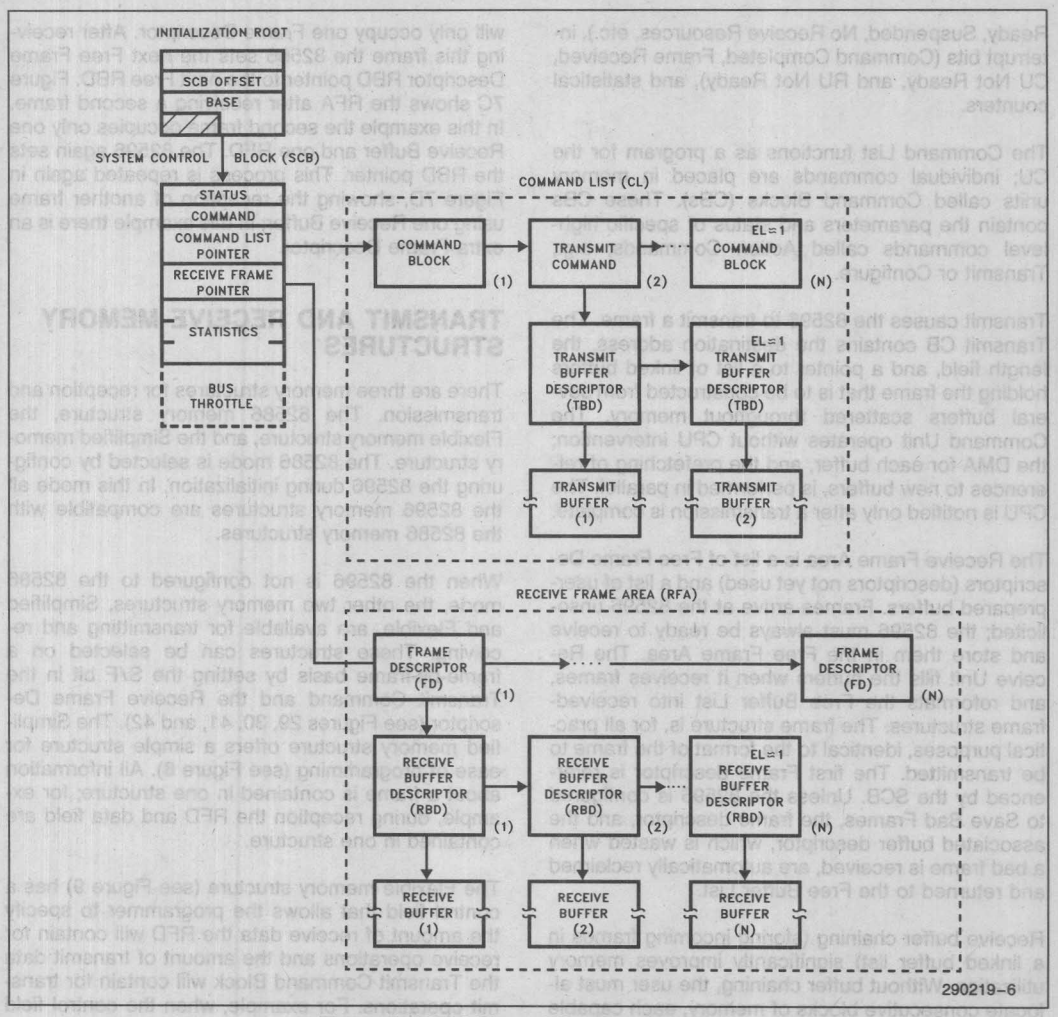


Figure 6. 82596 Shared Memory Structure

82596 SYSTEM MEMORY STRUCTURE

The Shared Memory structure consists of four parts: the Initialization Root, the System Control Block, the Command List, and the Receive Frame Area (see Figure 6).

The Initialization Root is in an established location known to the host CPU and the 82596 (00FFFFF6h). However, the CPU can establish the Initialization Root in another location by using the CPU Port access. This root is accessed during initialization, and points to the System Control Block.

The System Control Block serves as a bidirectional mail drop for the host CPU and the 82596 CU and RU. It is the central point through which the CPU and the 82596 exchange control and status information. The SCB has two areas. The first contains instructions from the CPU to the 82596. These include: control of the CU and RU (Start, Abort, Suspend, and Resume), a pointer to the list of CU commands, a pointer to the Receive Frame Area, a set of Interrupt Acknowledge bits, and the T-ON and T-OFF timers for the bus throttle. The second area contains status information the 82596 is sending to the CPU. Such as, the CU and RU states (Idle, Active

Ready, Suspended, No Receive Resources, etc.), interrupt bits (Command Completed, Frame Received, CU Not Ready, and RU Not Ready), and statistical counters.

The Command List functions as a program for the CU; individual commands are placed in memory units called Command Blocks (CBs). These CBs contain the parameters and status of specific high-level commands called Action Commands; e.g., Transmit or Configure.

Transmit causes the 82596 to transmit a frame. The Transmit CB contains the destination address, the length field, and a pointer to a list of linked buffers holding the frame that is to be constructed from several buffers scattered throughout memory. The Command Unit operates without CPU intervention; the DMA for each buffer, and the prefetching of references to new buffers, is performed in parallel. The CPU is notified only after a transmission is complete.

The Receive Frame Area is a list of Free Frame Descriptors (descriptors not yet used) and a list of user-prepared buffers. Frames arrive at the 82596 unsolicited; the 82596 must always be ready to receive and store them in the Free Frame Area. The Receive Unit fills the buffers when it receives frames, and reformats the Free Buffer List into received-frame structures. The frame structure is, for all practical purposes, identical to the format of the frame to be transmitted. The first Frame descriptor is referenced by the SCB. Unless the 82596 is configured to Save Bad Frames, the frame descriptor, and the associated buffer descriptor, which is wasted when a bad frame is received, are automatically reclaimed and returned to the Free Buffer List.

Receive buffer chaining (storing incoming frames in a linked buffer list) significantly improves memory utilization. Without buffer chaining, the user must allocate consecutive blocks of memory, each capable of containing a maximum frame (for Ethernet, 1518 bytes). Since an average frame is about 200 bytes, this is very inefficient. With buffer chaining, the user can allocate small buffers and the 82596 will only use those that are needed.

Figure 7 A–D illustrates how the 82596 uses the Receive Frame Area. Figure 7A shows an unused Receive Frame Area composed of Free Frame Descriptors and Free Receive Buffers prepared by the user. The SCB points to the first Frame Descriptor of the Frame Descriptor List. Figure 7B shows the same Receive Frame Area after receiving one frame. This first frame occupies two Receive Buffers and one Frame Descriptor—a valid received frame will only occupy one Frame Descriptor. After receiving this frame the 82596 sets the next Free Frame Descriptor RBD pointer to the next Free RBD. Figure 7C shows the RFA after receiving a second frame. In this example the second frame occupies only one Receive Buffer and one RFD. The 82596 again sets the RBD pointer. This process is repeated again in Figure 7D, showing the reception of another frame using one Receive Buffer; in this example there is an extra Frame Descriptor.

TRANSMIT AND RECEIVE MEMORY STRUCTURES

There are three memory structures for reception and transmission. The 82586 memory structure, the Flexible memory structure, and the Simplified memory structure. The 82586 mode is selected by configuring the 82596 during initialization. In this mode all the 82596 memory structures are compatible with the 82586 memory structures.

When the 82596 is not configured to the 82586 mode, the other two memory structures, Simplified and Flexible, are available for transmitting and receiving. These structures can be selected on a frame-by-frame basis by setting the S/F bit in the Transmit Command and the Receive Frame Descriptor (see Figures 29, 30, 41, and 42). The Simplified memory structure offers a simple structure for ease of programming (see Figure 8). All information about a frame is contained in one structure; for example, during reception the RFD and data field are contained in one structure.

The Flexible memory structure (see Figure 9) has a control field that allows the programmer to specify the amount of receive data the RFD will contain for receive operations and the amount of transmit data the Transmit Command Block will contain for transmit operations. For example, when the control field in the RFD is set to 20 bytes during a reception, the first 20 bytes of the data field are stored in the RFD (6 Bytes of Destination Address, 6 Bytes of Source Address, 2 Bytes of Length Field, and 6 Bytes of Data), and the remainder of the data field is stored in the Receive Data Buffers. This is useful for capturing frame headers when header information is contained in the data field. The header information can then be automatically stored in the RFD partitioned from the Receive Data Buffer.

The control field can also be used for the Transmit Command when the Flexible memory structure is used. The quantity of data field bytes to be transmitted from the Transmit Command Block is specified by the variable control field.

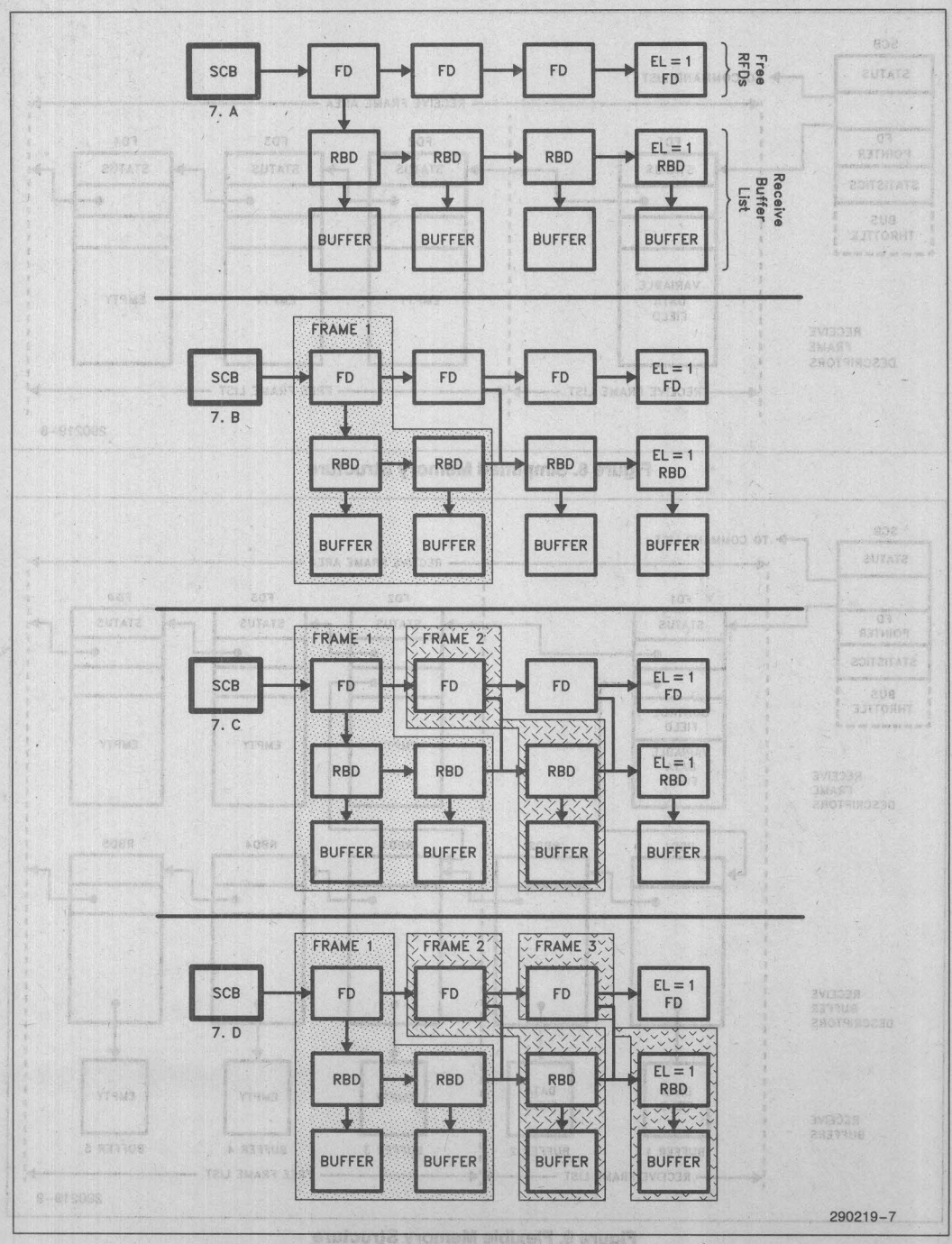


Figure 7. Frame Reception in the RFA

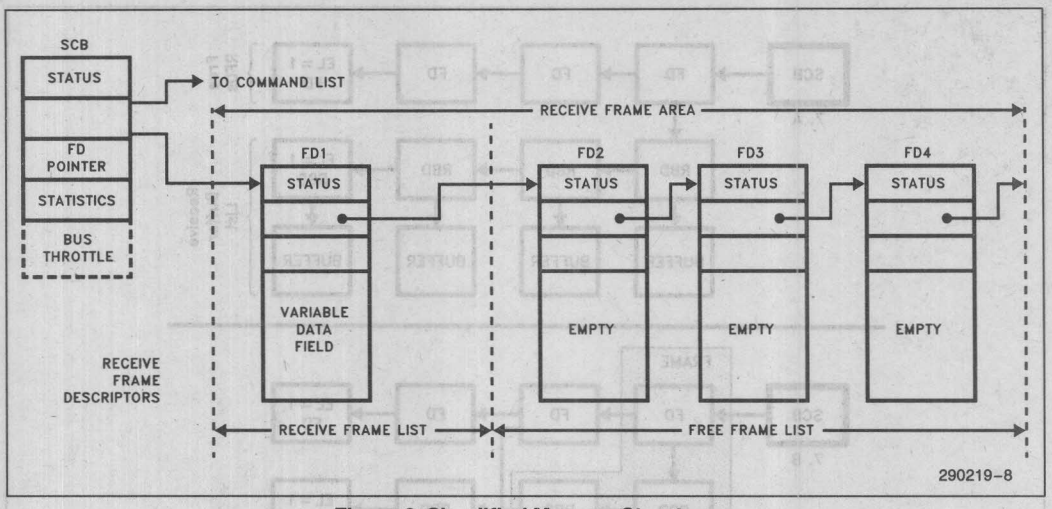


Figure 8. Simplified Memory Structure

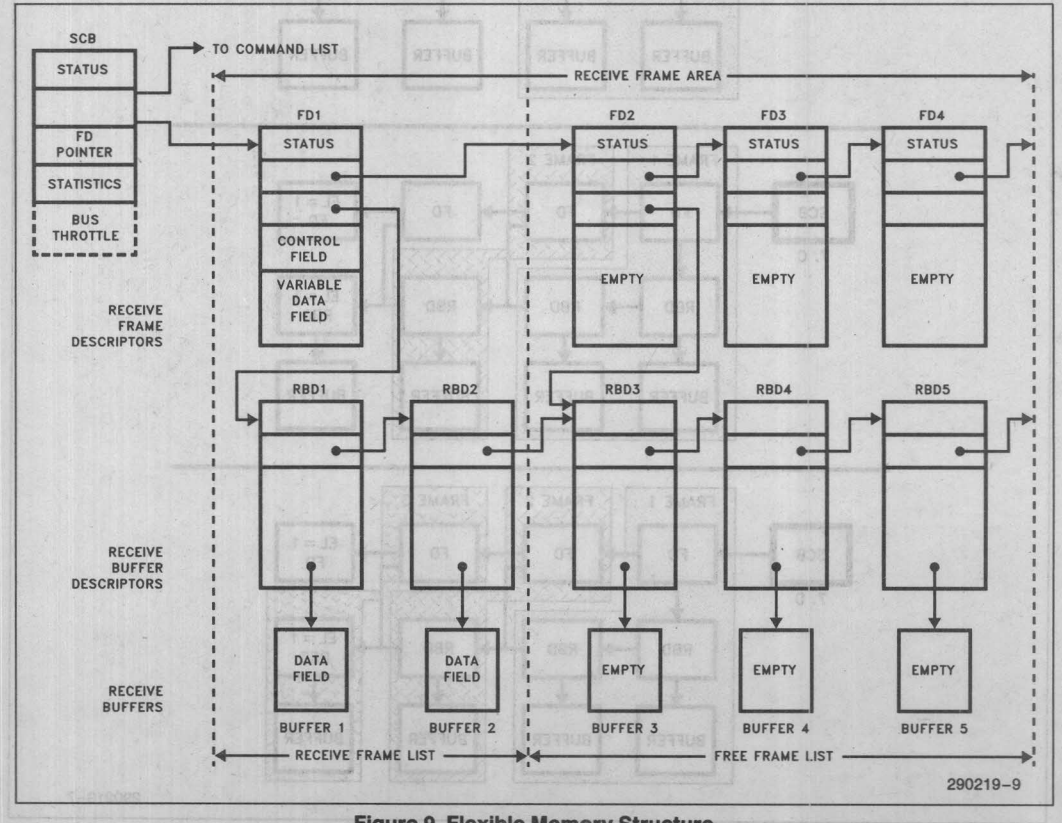


Figure 9. Flexible Memory Structure

TRANSMITTING FRAMES

The 82596 executes high-level Action Commands from the Command List in system memory. Action Commands are fetched and executed in parallel with the host CPU operation, thereby significantly improving system performance. The format of the Action Commands is shown in Figure 10. Figure 28 shows the 82586 mode, and Figures 29 and 30 shows the command formats of the Linear and 32-bit Segmented modes.

A single Transmit command contains, as part of the command-specific parameters, the destination address and length field of the transmitted frame and a pointer to buffer area in memory containing the data portion of the frame. The data field is contained in a memory data structure consisting of a buffer descriptor (BD) and a data buffer—or a linked list of buffer descriptors and buffers—as shown in Figure 11.

Multiple data buffers can be chained together using the BDs. Thus, a frame with a long data field can be transmitted using several (shorter) data buffers chained together. This chaining technique allows the system designer to develop efficient buffer management.

The 82596 automatically generates the preamble (alternating 1s and 0s) and start frame delimiter, fetches the destination address and length field from the Transmit command, inserts its unique address as the source address, fetches the data field specified by the Transmit command, and computes and appends the CRC to the end of the frame (see Figure 12). In the Linear and 32-bit Segmented mode the CRC can be optionally inserted on a frame-by-frame basis by setting the NC bit in the Transmit Command Block (see Figures 29 and 30).

The 82596 can be configured to generate two types of start and end frame delimiters—End of Carrier (EOC) or HDLC. In EOC mode the start frame delimiter is 10101011 and the end frame delimiter is indicated by the lack of a signal after the last bit of the frame check sequence field has been transmitted. In EOC mode the 82596 can be configured to extend short frames by adding pad bytes (7Eh) during transmission, according to the length field. In HDLC mode the 82596 will generate the 01111110 flag for the start and end frame delimiters, and do standard bit stuffing and stripping. Furthermore, the 82596 can be configured to pad frames shorter than the specified minimum frame length by appending the appropriate number of flags to the end of the frame.

When a collision occurs, the 82596 manages the jam, random wait, and retry processes, reinitializing DMA pointers without CPU intervention. Multiple frames can be sent by linking the appropriate number of Transmit commands together. This is particularly useful when transmitting a message larger than the maximum frame size (1518 bytes for Ethernet).

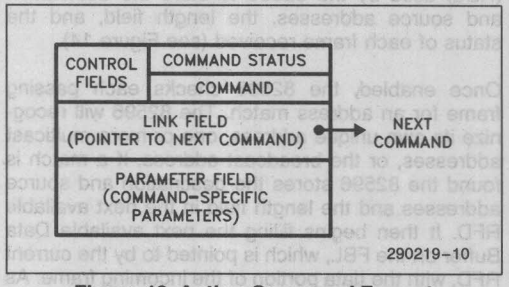


Figure 10. Action Command Format

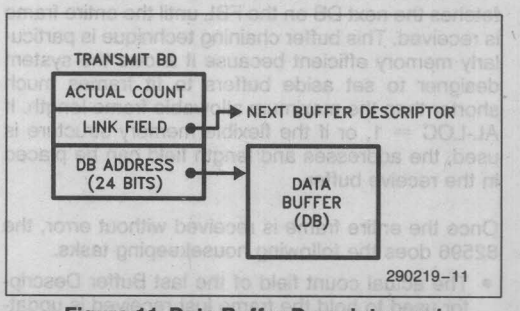


Figure 11. Data Buffer Descriptor and Data Buffer Structure

ers (see Figure 13). The individual Receive Frame Descriptors make up a Receive Descriptor List (RDL) used by the 82596 to store the destination and source addresses, the length field, and the status of each frame received (see Figure 14).

Once enabled, the 82596 checks each passing frame for an address match. The 82596 will recognize its own unique address, one or more multicast addresses, or the broadcast address. If a match is found the 82596 stores the destination and source addresses and the length field in the next available RFD. It then begins filling the next available Data Buffer on the FBL, which is pointed to by the current RFD, with the data portion of the incoming frame. As one Data Buffer is filled, the 82596 automatically fetches the next DB on the FBL until the entire frame is received. This buffer chaining technique is particularly memory efficient because it allows the system designer to set aside buffers to fit frames much shorter than the maximum allowable frame length. If AL-LOC = 1, or if the flexible memory structure is used, the addresses and length field can be placed in the receive buffer.

Once the entire frame is received without error, the 82596 does the following housekeeping tasks.

- The actual count field of the last Buffer Descriptor used to hold the frame just received is updated with the number of bytes stored in the associated Data Buffer.
- The next available Receive Frame Descriptor is fetched.
- The address of the next available Buffer Descriptor is written to the next available Receive Frame Descriptor.
- A frame received interrupt status bit is posted in the SCB.
- An interrupt is sent to the CPU.

If a frame error occurs, for example a CRC error, the 82596 automatically reinitializes its DMA pointers and reclaims any data buffers containing the bad

and asynchronous nature. It is particularly difficult to pinpoint a failure when it occurs. The 82596 has extensive diagnostic and network management functions that help improve reliability and testability. The 82596 reports on the following events after each frame is transmitted.

- Transmission successful.
- Transmission unsuccessful. Lost Carrier Sense.
- Transmission unsuccessful. Lost Clear to Send.
- Transmission unsuccessful. A DMA underrun occurred because the system bus did not keep up with the transmission.
- Transmission unsuccessful. The number of collisions exceeded the maximum allowed.
- Number of Collisions. The number of collisions experienced during the frame.
- Heartbeat Indicator. This indicates the presence of a heartbeat during the last Interframe Spacing (IFS) after transmission.

When configured to Save Bad Frames the 82596 checks each incoming frame and reports the following errors.

- CRC error. Incorrect CRC in a properly aligned frame.
- Alignment error. Incorrect CRC in a misaligned frame.
- Frame too short. The frame is shorter than the value configured for minimum frame length.
- Overrun. Part of the frame was not placed in memory because the system bus did not keep up with incoming data.
- Out of buffer. Part of the frame was discarded because of insufficient memory storage space.
- Receive collision. A collision was detected during reception.
- Length error. A frame not matching the frame length parameter was detected.

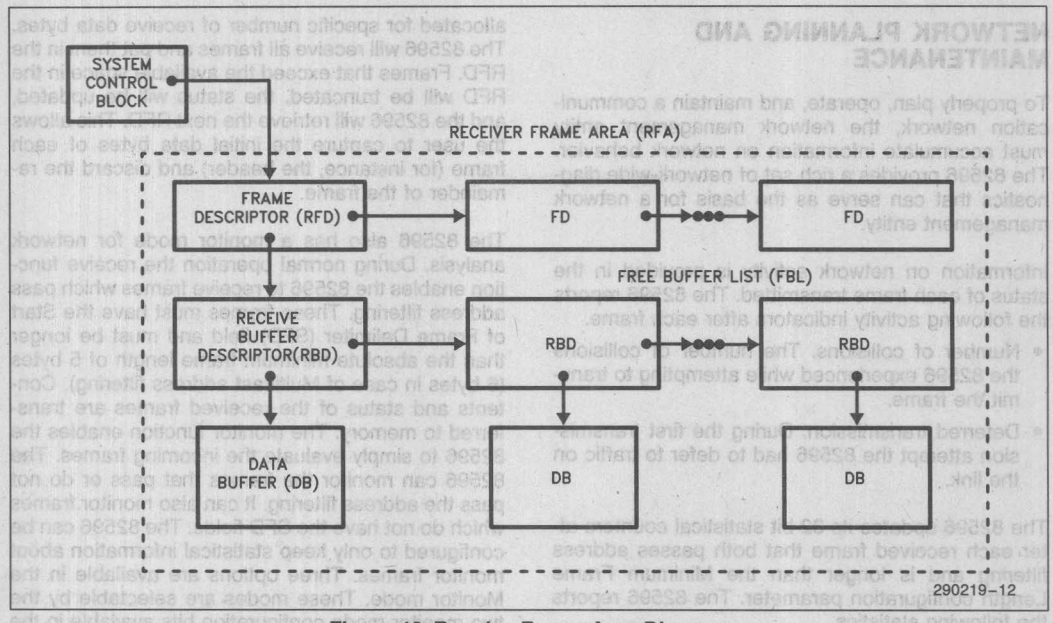


Figure 13. Receive Frame Area Diagram

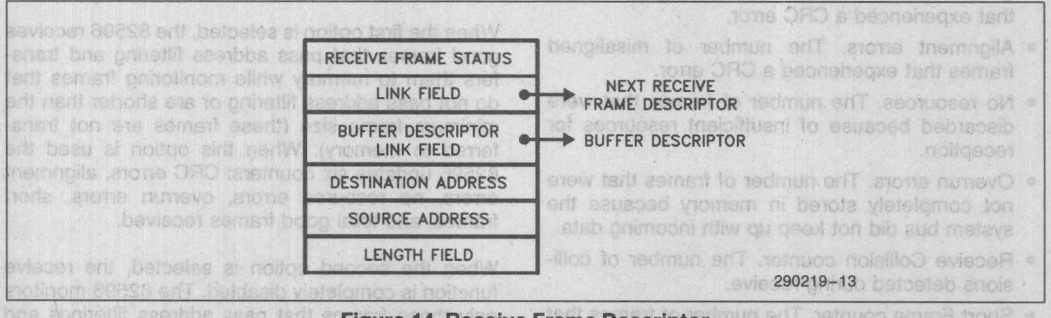


Figure 14. Receive Frame Descriptor

NETWORK PLANNING AND MAINTENANCE

To properly plan, operate, and maintain a communication network, the network management entity must accumulate information on network behavior. The 82596 provides a rich set of network-wide diagnostics that can serve as the basis for a network management entity.

Information on network activity is provided in the status of each frame transmitted. The 82596 reports the following activity indicators after each frame.

- Number of collisions. The number of collisions the 82596 experienced while attempting to transmit the frame.
- Deferred transmission. During the first transmission attempt the 82596 had to defer to traffic on the link.

The 82596 updates its 32-bit statistical counters after each received frame that both passes address filtering and is longer than the Minimum Frame Length configuration parameter. The 82596 reports the following statistics.

- CRC errors. The number of well-aligned frames that experienced a CRC error.
- Alignment errors. The number of misaligned frames that experienced a CRC error.
- No resources. The number of frames that were discarded because of insufficient resources for reception.
- Overrun errors. The number of frames that were not completely stored in memory because the system bus did not keep up with incoming data.
- Receive Collision counter. The number of collisions detected during receive.
- Short Frame counter. The number of frames that were discarded because they were shorter than the configured minimum frame length.

The 82596 can be configured to Promiscuous mode. In this mode it captures all frames transmitted on the network without checking the Destination Address. This is useful when implementing a monitoring station to capture all frames for analysis.

A useful method of capturing frame headers is to use the Simplified memory mode, configure the 82596 to Save Bad Frames, and configure the 82596 to Promiscuous mode with space in the RFD allocated for specific number of receive data bytes. The 82596 will receive all frames and put them in the RFD. Frames that exceed the available space in the RFD will be truncated, the status will be updated, and the 82596 will retrieve the next RFD. This allows the user to capture the initial data bytes of each frame (for instance, the header) and discard the remainder of the frame.

The 82596 also has a monitor mode for network analysis. During normal operation the receive function enables the 82596 to receive frames which pass address filtering. These frames must have the Start of Frame Delimiter (SFD) field and must be longer than the absolute minimum frame length of 5 bytes (6 bytes in case of Multicast address filtering). Contents and status of the received frames are transferred to memory. The monitor function enables the 82596 to simply evaluate the incoming frames. The 82596 can monitor the frames that pass or do not pass the address filtering. It can also monitor frames which do not have the SFD fields. The 82596 can be configured to only keep statistical information about monitor frames. Three options are available in the Monitor mode. These modes are selectable by the two monitor mode configuration bits available in the configuration command.

When the first option is selected, the 82596 receives good frames that pass address filtering and transfers them to memory while monitoring frames that do not pass address filtering or are shorter than the minimum frame size (these frames are not transferred to memory). When this option is used the 82596 updates six counters: CRC errors, alignment errors, no resource errors, overrun errors, short frames, and total good frames received.

When the second option is selected, the receive function is completely disabled. The 82596 monitors only those frames that pass address filterings and meet the minimum frame length requirement. When this option is used the 82596 updates six counters: CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected, and total good frames.

When the third option is selected, the receive function is completely disabled. The 82596 monitors all frames, including frames that do not have a Start Frame Delimiter. When this option is used the 82596 updates six counter (CRC errors, alignment errors, total frames (good and bad), short frames, collisions detected, and total good frames.

STATION DIAGNOSTICS AND SELF-TEST

The 82596 provides a large set of diagnostic and network management functions. These include internal and external loopback and time domain reflectometry for locating fault points in the network cable. The 82596 ensures software reliability by dumping the contents of the 82596 internal registers into system memory. The 82596 has a self-test mode that enables it to run an internal self-test and place the results in system memory.

82586 SOFTWARE COMPATIBILITY

The 82596 has a software-compatible state in which all its memory structures are compatible with the 82586 memory structure. This includes all the Action Commands, the Receive Frame Area (including the RFD, Buffer Descriptors, and Data Buffers), the System Control Block, and the initialization procedures. There are two minor differences between the 82596 in the 82586-Compatible memory structure and the 82586.

- When the internal and external loopback bits in the Configure command are set to 11 the 82596 is in external loopback and the $\overline{\text{LPBK}}$ pin is activated; in the 82586 this situation would produce internal loopback.
- During a Dump command both the 82596 and 82586 dump the same number of bytes; however, the data format is different.

INITIALIZING THE 82596

A Reset command is issued to the 82596 to prepare it for normal operation. The 82596 is initialized through two data structures that are addressed by two pointers, the System Configuration Pointer (SCP) and the Intermediate System Configuration Pointer (ISCP). The initialization procedure begins when a Channel Attention signal is asserted after RESET. The 82596 uses the address of the double word that contains the SCP as a default—00FFFFF4h. Before the CA signal is asserted this default address can be changed to any other available address by asserting the $\overline{\text{PORT}}$ pin and providing the desired address over the $D_{31}-D_4$ pins of the address bus. Pins D_3-D_0 must be 0010; i.e., any alternative address must be aligned to 16 byte boundaries. All addresses sent to the 82596 must be word aligned, which means that all pointers and memory structures must start on an even address (A_0 = zero).

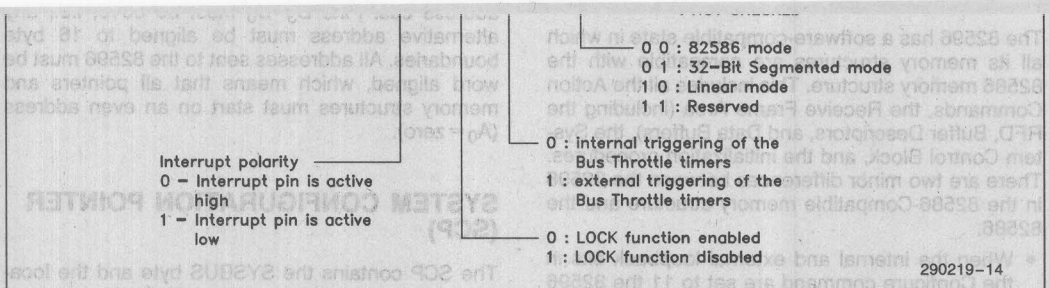
SYSTEM CONFIGURATION POINTER (SCP)

The SCP contains the SYSBUS byte and the location of the next structure of the initialization process, the ISCP. The following parameters are selected in the SYSBUS.

- The 82596 operation mode.
- The Bus Throttle timer triggering method.
- Lock enabled.
- Interrupt polarity.

Byte ordering is determined by the LE/$\overline{\text{BE}}$ pin. LE/$\overline{\text{BE}}$ = 1 selects little endian byte ordering and LE/$\overline{\text{BE}}$ = 0 selects big endian byte ordering.

NOTE:
In the following, X indicates a bit not checked in 82586 mode. This bit must be set to 0 in all other modes.



ISCP ADDRESS— The physical address of the ISCP. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 15. The System Configuration Pointer

Writing the Sysbus

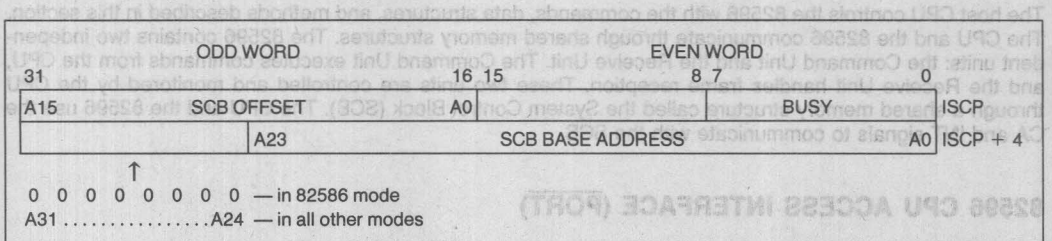
When writing the Sysbus byte it is important to pay attention to the byte order.

- When a Little Endian processor is used, the Sysbus byte is located at byte address 00FFFFF6h (or address $n+2$ if an alternative SCP address n was programmed).
- When a processor using Big Endian byte ordering is used, the SYSBUS, alternative SCP, and ISCP addresses will be different.
  - The Sysbus byte is located at 00FFFFF5h.
  - If an alternative SCP address is programmed, the SYSBUS byte should be at byte address $n+1$.

INTERMEDIATE SYSTEM CONFIGURATION POINTER (ISCP)

The ISCP indicates the location of the System Control Block. Often the SCP is in ROM and the ISCP is in RAM. The CPU loads the SCB address (or an equivalent data structure) into the ISCP and asserts CA. This Channel Attention signal causes the 82596 to begin its initialization procedure and to get the SCB address from the ISCP and SCP. In 82586 and 32-bit Segmented modes the SCP base address is also the base address of all Command Blocks, Frame Descriptors, and Buffer Descriptors (but not buffers). All these data structures must reside in one 64-kB segment; however, in Linear mode no such limitation is imposed.

The following diagram illustrates the ISCP format.

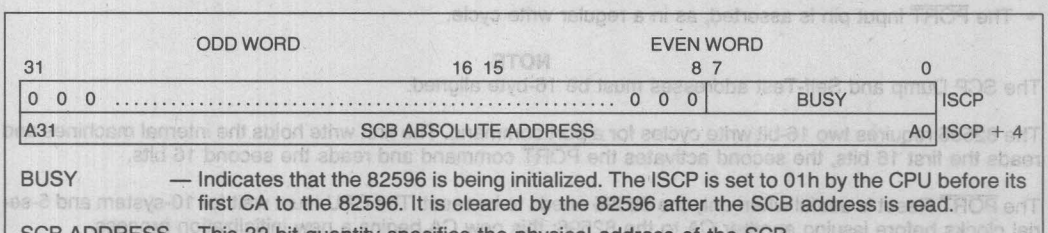


BUSY — Indicates that the 82596 is being initialized. The CPU sets the ISCP to 01h before it gives the first CA to the 82596. The ISCP is cleared by the 82596 after the SCB base and offset are read. Note that the most significant byte of the first word of the ISCP is not modified when BUSY is cleared.

SCB OFFSET— This 16-bit quantity specifies the offset portion of the address of the SCB.

SCB BASE — Specifies the base portion of the address of the SCB. The base of SCB is also the base of all 82596 Command Blocks, Frame Descriptors and Buffer Descriptors. In the 82586 mode, bits A31–A24 are considered to be zero.

Figure 16. The Intermediate System Configuration Pointer—82586 and 32-Bit Segmented Modes



BUSY — Indicates that the 82596 is being initialized. The ISCP is set to 01h by the CPU before its first CA to the 82596. It is cleared by the 82596 after the SCB address is read.

SCB ADDRESS— This 32-bit quantity specifies the physical address of the SCB.

Figure 17. The Intermediate System Configuration Pointer—Linear Mode.

INITIALIZATION PROCESS

The CPU sets up the SCP, ISCP, and the SCB structures, and, if desired, an alternative SCP address. It also sets BUSY to 01h. The 82596 is initialized when a Channel Attention signal follows a Reset signal, causing the 82596 to access the System Configuration Pointer. The sysbus byte, the operational mode, the bus throttle timer triggering method, the interrupt polarity, and the state of $\overline{LOCK}$ are read. After reset the bus throttle

timers are essentially disabled—the T-ON value is infinite, the T-OFF value is zero. After the SCP is read, the 82596 reads the ISCP and saves the SCB address. In 82586 and 32-bit Segmented modes this address is represented as a base address plus the offset (this base address is also the base address of all the control blocks). In Linear mode the base address is also an absolute address. The 82596 clears BUSY, sets CX and CNR to equal 1 in the SCB, clears the SCB command word, sends an interrupt to the CPU, and awaits another Channel Attention signal. RESET configures the 82596 to its default state before CA is asserted.

CONTROLLING THE 82596DX/SX

The host CPU controls the 82596 with the commands, data structures, and methods described in this section. The CPU and the 82596 communicate through shared memory structures. The 82596 contains two independent units: the Command Unit and the Receive Unit. The Command Unit executes commands from the CPU, and the Receive Unit handles frame reception. These two units are controlled and monitored by the CPU through a shared memory structure called the System Control Block (SCB). The CPU and the 82596 use the CA and INT signals to communicate with the SCB.

82596 CPU ACCESS INTERFACE ($\overline{PORT}$)

The 82596 has a CPU access interface that allows the host CPU to do four things.

- Write an alternative System Configuration Pointer address.
- Write an alternative Dump area pointer and perform Dump.
- Execute a software reset.
- Execute a self-test.

The following events initiate the CPU access state.

- Presence of an address on the D_{31}–D_4 data bus pins.
- The D_3–D_0 pins are used to select one of the four functions.
- The $\overline{PORT}$ input pin is asserted, as in a regular write cycle.

NOTE

The SCP Dump and Self-Test addresses must be 16-byte aligned.

The 82596 requires two 16-bit write cycles for a port command. The first write holds the internal machines and reads the first 16 bits, the second activates the $\overline{PORT}$ command and reads the second 16 bits.

The $\overline{PORT}$ Reset is useful when only the 82596 needs to be reset. The CPU must wait for 10-system and 5-serial clocks before issuing another CA to the 82596; this new CA begins a new initialization process.

The Dump function is useful for troubleshooting No Response problems. If the chip is in a No Response state, the $\overline{PORT}$ Dump operation can be executed and a $\overline{PORT}$ Reset can be used to reinitialize the 82596 without disturbing the rest of the system.

The Self-Test function can be used for board testing; the 82596 will execute a self-test and write the results to memory.

Table 2. $\overline{PORT}$ Function Selection

	D_{31} ..... D_4 ..... D_0						
Function	**Addresses and Results**			D_3	D_2	D_1	D_0
Reset	A31	Don't Care	A4	0	0	0	0
Self-Test	A31	Self-Test Results Address	A4	0	0	0	1
SCP	A31	Alternative SCP Address	A4	0	0	1	0
Dump	A31	Dump Area Pointer	A4	0	0	1	1

MEMORY ADDRESSING FORMATS

The 82596 accesses memory by 32-bit addresses. There are two types of 32-bit addresses: linear and segmented. The type of address used depends on the 82596 operating mode and the type of memory structure it is addressing. The 82596 has three operating modes.

- 82586 Mode
  - A Linear address is a single 24-bit entity. Address pins A_{31}–A_{24} are always zero.
  - A Segmented address uses a 24-bit base and a 16-bit offset.
- 32-bit Segmented Mode
  - A Linear address is a single 32-bit entity.
  - A Segmented address uses a 32-bit base and a 16-bit offset.

NOTE

In the previous two memory addressing modes, each command header (CB, TBD, RFD, RBD, and SCB) must wholly reside within one segment. If the 82596 encounters a memory structure that does not follow this restriction, the 82596 will fetch the next contiguous location in memory (beyond the segment).

- Linear Mode
  - A Linear address is a single 32-bit entity.
  - There are no Segmented addresses.

Linear addresses are primarily used to address transmit and receive data buffers. In the 82586 and 32-bit Segmented modes, segmented addresses (base plus offset) are used for all Command Blocks, Buffer Descriptors, Frame Descriptors, and System Control Blocks. When using Segmented addresses, only the offset portion of the entity being addressed is specified in the block. The base for all offsets is the same—that of the SCB.

COMMAND UNIT (CU)

The Command Unit is the logical unit that executes Action Commands from a list of commands very similar to a CPU program. A Command Block is associated with each Action Command. The CU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The CU is not executing a command and is not associated with a CB on the list. This is the initial state.
- **Suspended.** The CU is not executing a command; however, it is associated with a CB on the list. The suspend state can only be reached if the CPU forces it through the SCB or sets the suspend bit in the RFD.
- **Active.** The CU is executing an Action Command and pointing to its CB.

The CPU can affect CU operation in two ways: by issuing a CU Control Command or by setting bits in the Command word of the Action Command.

RECEIVE UNIT (RU)

The Receive Unit is the logical unit that receives frames and stores them in memory. The RU is modeled as a logical machine that takes, at any given time, one of the following states.

- **Idle.** The RU has no memory resources and is discarding incoming frames. This is the initial state.
- **No Resources.** The RU has no memory resources and is discarding incoming frames. This state differs from Idle in that the RU accumulates statistics on the number of discarded frames.
- **Suspended.** The RU has memory available for storing frames, but is discarding them. The suspend state can only be reached if the CPU forces it through the SCB or sets the suspend bit in the RFD.
- **Ready.** The RU has memory available and is storing incoming frames.

The CPU can affect RU operation in three ways: by issuing an RU Control Command, by setting bits in the Frame Descriptor Command word of the frame being received, or by setting the EL bit of the current buffer's Buffer Descriptor.

SYSTEM CONTROL BLOCK (SCB)

The SCB is a memory block that plays a major role in communications between the CPU and the 82596. Such communications include the following.

- Commands issued by the CPU
- Status reported by the 82596

Control commands are sent to the 82596 by writing them into the SCB and then asserting CA. The 82596 examines the command, performs the required action, and then clears the SCB command word. Control commands perform the following types of tasks.

- Operation of the Command Unit (CU). The SCB controls the CU by specifying the address of the Command Block List (CBL) and by starting, suspending, resuming, or aborting execution of CBL commands.

- The Command Unit becomes inactive.
- The Receive Unit becomes not ready.
- The status of the Command Unit.
- The status of the Receive Unit.
- Status reports from the 82596 regarding reception of corrupted frames.

Events can be cleared only by CPU acknowledgment. If some events are not acknowledged by the ACK field the Interrupt signal (INT) will be reissued after Channel Attention (CA) is processed. Furthermore, if a new event occurs while an interrupt is set, the interrupt is temporarily cleared to trigger edge-triggered interrupt controllers.

The CPU uses the Channel Attention line to cause the 82596 to examine the SCB. This signal is trailing-edge triggered—the 82596 latches CA on the trailing edge. The latch is cleared by the 82596 before the SCB control command is read.

31 ODD WORD 16	15 EVEN WORD 0	
ACK X CUC R RUC X X X X	STAT 0 CUS 0 RUS 0 0 0 0	SCB
RFA OFFSET	CBL OFFSET	SCB + 4
ALIGNMENT ERRORS	CRC ERRORS	SCB + 8
OVERRUN ERRORS	RESOURCE ERRORS	SCB + 12

Figure 18. SCB—82586 Mode

31 ODD WORD 16	15 EVEN WORD 0	
ACK 0 CUC R RUC 0 0 0 0	STAT 0 CUS T RUS 0 0 0	SCB
RFA OFFSET	CBL OFFSET	SCB + 4
CRC ERRORS		SCB + 8
ALIGNMENT ERRORS		SCB + 12
RESOURCE ERRORS (*)		SCB + 16
OVERRUN ERRORS (*)		SCB + 20
RCVCDT ERRORS (*)		SCB + 24
SHORT FRAME ERRORS		SCB + 28
T-ON TIMER	T-OFF TIMER	SCB + 32

*In MONITOR mode these counters change function

Figure 19. SCB—32-Bit Segmented Mode

31 ODD WORD 16	15 EVEN WORD 0	
ACK 0 CUC R RUC 0 0 0 0	STAT 0 CUS T RUS 0 0 0 0	SCB
COMMAND BLOCK ADDRESS		SCB + 4
RECEIVE FRAME AREA ADDRESS		SCB + 8
CRC ERRORS		SCB + 12
ALIGNMENT ERRORS		SCB + 16
RESOURCE ERRORS (*)		SCB + 20
OVERRUN ERRORS (*)		SCB + 24
RCVCDT ERRORS (*)		SCB + 28
SHORT FRAME ERRORS		SCB + 32
T-ON TIMER	T-OFF TIMER	SCB + 36

*In MONITOR mode these counters change function

Figure 20. SCB—Linear Mode

Command Word

31 16						
ACK	0	CUC	R	RUC	0 0 0 0	SCB + 2

These bits specifiy the action to be performed as a result of a CA. This word is set by the CPU and cleared by the 82596. Defined bits are:

Bit 31 ACK-CX — Acknowledges that the CU completed an Action Command.

Bit 30 ACK-FR — Acknowledges that the RU received a frame.

Bit 29 ACK-CNA — Acknowledges that the Command Unit became not active.

Bit 28 ACK-RNR — Acknowledges that the Receive Unit became not ready.

Bits 24–26 CUC — (3 bits) This field contains the command to the Command Unit. Valid values are:

- 0 — NOP (does not affect current state of the unit).
- 1 — Start execution of the first command on the CBL. If a command is executing, complete it before starting the new CBL. The beginning of the CBL is in CBL OFFSET (address).
- 2 — Resume the operation of the Command Unit by executing the next command. This operation assumes that the Command Unit has been previously suspended.
- 3 — Suspend execution of commands on CBL after current command is complete.
- 4 — Abort current command immediately.
- 5 — Loads the Bus Throttle timers so they will be initialized with their new values after the active timer (T-ON or T-OFF) reaches Terminal Count. If no timer is active new values will be loaded immediately. This command is not valid in 82586 mode.
- 6 — Loads and immediately restarts the Bus Throttle timers with their new values. This command is not valid in 82586 mode.
- 7 — Reserved.

Bits 20–22 RUC — (3 bits) This field contains the command to the Receive Unit. Valid values are:

0 — NOP (does not alter current state of unit).

1 — Start reception of frames. The beginning of the RFA is contained in the RFA OFFSET (address). If a frame is being received complete reception before starting.

2 — Resume frame reception (only when in suspended state).

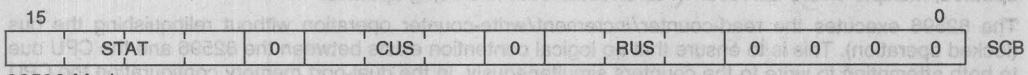
3 — Suspend frame reception. If a frame is being received complete its reception before suspending.

4 — Abort receiver operation immediately.

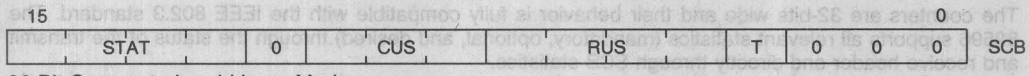
5–7 — Reserved.

Bit 23 RESET — Reset chip (logically the same as hardware RESET).

Status Word



82586 Mode



32-Bit Segmented and Linear Modes

Indicates the status of the 82596. This word is modified only by the 82596. Defined bits are:

Bit 15 CX — The CU finished executing a command with its I (interrupt) bit set.

Bit 14 FR — The RU finished receiving a frame.

Bit 13 CNA — The Command Unit left the Active state.

Bit 12 RNR — The Receive Unit left the Ready state.

Bits 8–10 CUS — (3 bits) This field contains the status of the command unit. Valid values are:

0 — Idle

1 — Suspended

2 — Active

3–7 — Not used

Bits 4–7 RUS — This field contains the status of the receive unit. Valid values are:

0h (0000) — Idle

1h (0001) — Suspended

2h (0010) — No resources. This bit indicates both no resources due to lack of RFDs in the RDL and no resources due to lack of RBDs in the FBL.

4h (0100) — Ready

8h (1000) — No more RBDs (not in the 82586 mode).

Ah (1010) — No resources due to no more RBDs. (Not in the 82586 mode.)

No other combinations are allowed.

Bit 3 T — Bus Throttle timers loaded (not in 82586 mode).

SCB OFFSET ADDRESSES

CBL Offset (Address)

In 82586 and 32-bit Segmented modes this 16-bit quantity indicates the offset portion of the address for the first Command Block on the CBL. In Linear mode it is a 32-bit linear address for the first Command Block on the CBL. It is accessed only if CUC equals Start.

updated; multiple errors will result in all affected counters being updated.

- The 82596 executes the read-counter/increment/write-counter operation without relinquishing the bus (locked operation). This is to ensure that no logical contention exists between the 82596 and the CPU due to both attempting to write to the counters simultaneously. In the dual-port memory configuration the CPU should not execute any write operation to a counter if $\overline{LOCK}$ is asserted.
- The counters are 32-bits wide and their behavior is fully compatible with the IEEE 802.3 standard. The 82596 supports all relevant statistics (mandatory, optional, and desired) through the status of the transmit and receive header and directly through SCB statistics.

CRCERRS

This 32-bit quantity contains the number of aligned frames discarded because of a CRC error. This counter is updated, if needed, regardless of the RU state.

ALNERRS

This 32-bit quantity contains the number of frames that both are misaligned (i.e., where $\overline{CRS}$ deasserts on a nonoctet boundary) and contain a CRC error. The counter is updated, if needed, regardless of the RU state.

SHRTFRM

This 32-bit quantity contains the number of received frames shorter than the minimum frame length.

The last three counters change function in monitor mode.

RSCERRS

This 32-bit quantity contains the number of good frames discarded because there were no resources to contain them. Frames intended for a host whose RU is in the No Receive Resources state, fall into this category. This counter is updated only if the RU is in the No Resources state. When in Moniitor mode, this counter counts the total number of frames.

OVRNERRS

This 32-bit quantity contains the number of frames known to be lost because the local system bus was not available. If the traffic problem lasts longer than the duration of one frame, the frames that follow the first are lost without an indicator, and they are not counted. This counter is updated, if needed, regardless of the RU state.

RCVCDT

This 32-bit quantity contains the number of collisions detected during frame reception. In Monitor mode this counter counts the total number of good frames.

ACTION COMMANDS AND OPERATING MODES

This section lists all the Action Commands of the Command Unit Command Block List (CBL). Each command contains the Command field, the Status and Control fields, the link to the next Action Command, and any command-specific parameters. There are three basic types of action commands: 82596 Configuration and Setup, Transmission, and Diagnostics. The following is a list of the actual commands.

- NOP
- Individual Address Setup
- Configure
- MC Setup
- Transmit
- TDR
- Dump
- Diagnose

The 82596 has three addressing modes. In the 82586 mode all the Action Commands look exactly like those of the 82586.

- **82586 Mode.** The 82596 software and memory structure is compatible with the 82586.
- **32-Bit Segmented Mode.** The 82596 can access the entire system memory and use the two new memory structures—Simplified and Flexible—while still using the segmented approach. This does not require any significant changes to existing software.
- **Linear Mode.** The 82596 operates in a flat, linear, 4 gigabyte memory space without segmentation. It can also use the two new memory structures.

In the 32-bit Segmented mode there are some differences between the 82596 and 82586 action commands, mainly in programming and activating new 82596 features. Those bits marked "don't care" in the compatible mode are not checked; however, we strongly recommend that those bits all be zeroes; this will allow future enchancements and extensions.

In the Linear mode all of the address offsets become 32-bit address pointers. All new 82596 features are accessible in this mode, and all bits previously marked "don't care" must be zeroes.

The Action Commands, and all other 82596 memory structures, must begin on even byte boundaries, i.e., they must be word aligned.

NOP

This command results in no action by the 82596 except for those performed in the normal command processing. It is used to manipulate the CBL manipulation. The format of the NOP command is shown in Figure 21.

NOP—82586 and 32-Bit Segmented Modes

31 ODD WORD																16	15 EVEN WORD														0	
EL	S	I	X	X	X	X	X	X	X	X	X	X	0	0	0	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	A15															A0	4

NOP—Linear Mode

31 ODD WORD																16	15 EVEN WORD														0	
EL	S	I	0	0	0	0	0	0	0	0	0	0	0	0	0	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A31																															A0	4

Figure 21

where:

LINK POINTER — In the 82586 or 32-bit Segmented modes this is a 16-bit offset to the next Command Block. In the Linear mode this is the 32-bit address of the next Command Block.

EL — If set, this bit indicates that this command block is the last on the CBL.

S — If set to one, suspend the CU upon completion of this CB.

I — If set to one, the 82596 will generate an interrupt after execution of the command is complete. If I is not set to one, the CX bit will not be set.

CMD (bits 16–18) — The NOP command. Value: 0h.

Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).

C — This bit indicates the execution status of the command. The CPU initially resets it to zero when the Command Block is placed on the CBL. Following a command Completion, the 82596 will set it to one.

B — This bit indicates that the 82596 is currently executing the NOP command. It is initially reset to zero by the CPU. The 82596 sets it to one when execution begins and to zero when execution is completed. This bit is also set when the 82596 prefetches the command.

NOTE:
The C and B bits are modified in one operation.

OK — Indicates that the command was executed without error. If set to one no error occurred (command executed OK). If zero an error occur.

INDIVIDUAL ADDRESS SETUP

This command is used to load the 82596 with the Individual Address. This address is used by the 82596 for inserting the Source Address during transmission and recognizing the Destination Address during reception. After RESET, and prior to Individual Address Setup Command execution, the 82596 assumes the Broadcast Address is the Individual Address in all aspects, i.e.:

- This will be the Individual Address Match reference.
- This will be the Source Address of a transmitted frame (for AL-LOC=0 mode only).

The format of the Individual Address Setup command is shown in Figure 22.

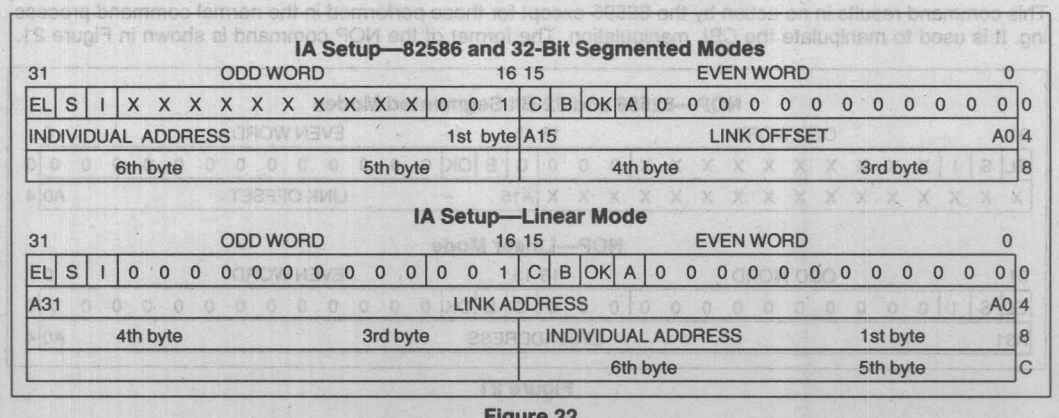


Figure 22

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (zero in the 32-bit Segmented and Linear modes).

CMD (bits 16–18) — The Address Setup command. Value: 1h.

INDIVIDUAL ADDRESS — The individual address of the node, 0 to 6 bytes long.

The least significant bit of the Individual Address must be zero for Ethernet (see the Command Structure). However, no enforcement of 0 is provided by the 82596. Thus, an Individual Address with 1 as its least significant bit is a valid Individual Address in all aspects.

The default address length is 6 bytes long, as in 802.3. If a different length is used the IA Setup command should be executed after the Configure command.

CONFIGURE

The Configure command loads the 82596 with its operating parameters. It allows changing some of the parameters by specifying a byte count less than the maximum number of configuration bytes (12 in the 82586 mode, 16 in the 32-Bit Segmented and Linear modes). The 82596 configuration depends on its mode of operation.

- In the 82586 mode the maximum number of configuration bytes is 12. Any number larger than 12 will be reduced to 12 and any number less than 4 will be increased to 4.
- The additional features of the serial side are disabled in the 82586 mode.
- In both the 32-Bit Segmented and Linear modes there are four additional configuration bytes, which hold parameters for additional 82596 features. If these parameters are not accessed, the 82596 will follow their default values.
- For more detailed information refer to the 82596 User's Manual.

The format of the Configure command is shown in Figures 23, 24, and 25.

31 ODD WORD 16	15 EVEN WORD 0	
EL S I X X X X X X X X X X 0 1 0	C B OK A 0 0 0 0 0 0 0 0 0 0 0 0	0
Byte 1 / Byte 0	A15 LINK OFFSET A0	4
Byte 5 / Byte 4	Byte 3 / Byte 2	8
Byte 9 / Byte 8	Byte 7 / Byte 6	1
X X X X X X X X X X X X X X X X	X X X X X X X X / Byte 10	1

Figure 23. CONFIGURE—82586 Mode

31 ODD WORD 16	15 EVEN WORD 0	
EL S I 0 0 0 0 0 0 0 0 0 0 0 1 0	C B OK A 0 0 0 0 0 0 0 0 0 0 0 0	0
Byte 1 / Byte 0	A15 LINK OFFSET A0	4
Byte 5 / Byte 4	Byte 3 / Byte 2	8
Byte 9 / Byte 8	Byte 7 / Byte 6	12
Byte 13 / Byte 12	Byte 11 / Byte10	16

Figure 24. CONFIGURE—32-Bit Segmented Mode

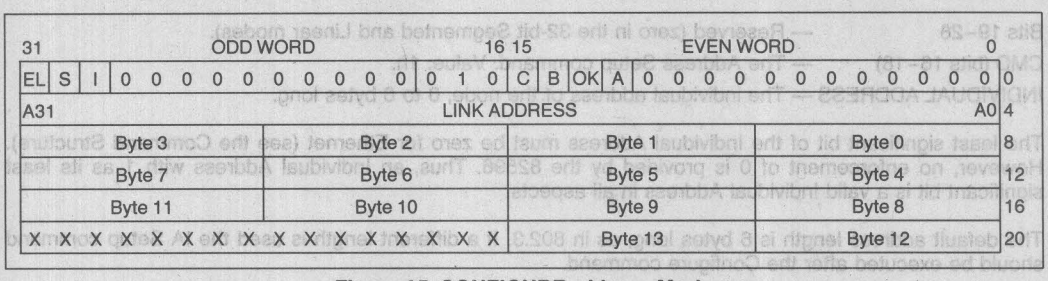


Figure 25. CONFIGURE—Linear Mode

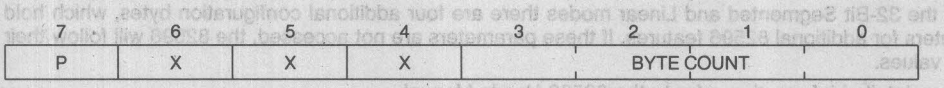
LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If 1, then the command was aborted and if necessary it should be repeated. If this bit is 0, the command was not aborted.

Bits 19–28 — Reserved (zero in the 32-Bit Segmented and Linear Modes)

CMD (bits 16–18) — The CONFIGURE command. Value: 2h.

The interpretation of the fields follows:

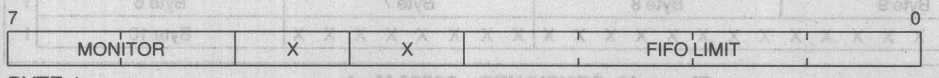


BYTE 0

BYTE CNT (Bits 0–3) — Byte Count. Number of bytes, including this one, that hold parameters to be configured.

PREFETCHED (Bit 7) — Enable the 82596 to write the prefetched bit in all prefetch RBDs.

NOTE:
The P bit is valid only in the new memory structure modes. In 82586 mode this bit is disabled (i.e., no prefetched mark).

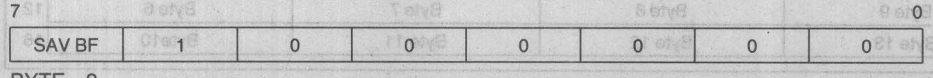


BYTE 1

FIFO Limit (Bits 0–3) — FIFO limit.

MONITOR# (Bits 6–7) — Receive monitor options. If the Byte Count of the configure command is less than 12 bytes then these Monitor bits are ignored.

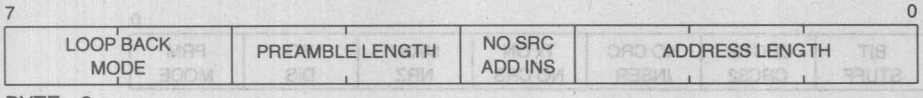
DEFAULT: C8h



BYTE 2

SAV BF (Bit 7) — 0—Received bad frames are not saved in the memory.
1—Received bad frames are saved in the memory.

DEFAULT: 40h

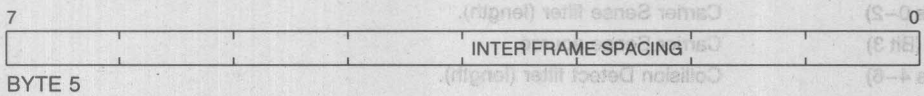


BYTE 3

ADR LEN (Bits 0–2)	Address length (any kind).
NO SCR ADD INS (Bit 3)	No Source Address Insertion. In the 82586 this bit is called AL LOC.
PREAM LEN (Bits 4–5)	Preamble length.
LP BCK MODE (Bits 6–7)	Loopback mode.
DEFAULT: 26h	

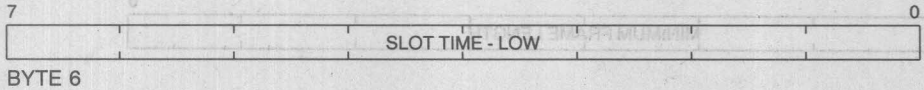
BYTE 4

LIN PRIO (Bits 0–2)	Linear Priority.
EXP PRIO (Bits 4–6)	Exponential Priority.
BOF METD (Bit 7)	Exponential Backoff method.
DEFAULT: 00h	



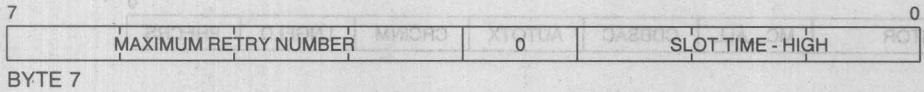
BYTE 5

INTERFRAME SPACING	Interframe spacing.
DEFAULT: 60h	



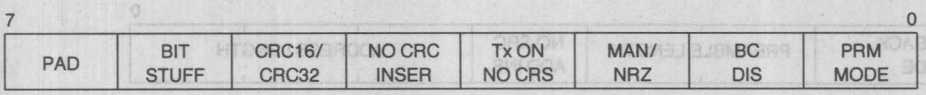
BYTE 6

SLOT TIME (L)	Slot time, low byte.
DEFAULT: 00h	



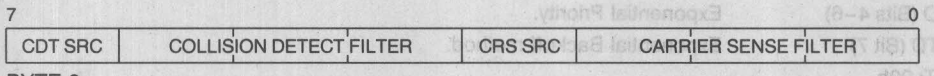
BYTE 7

SLOT TIME (H) (Bits 0–2)	Slot time, high part.
RETRY NUM (Bits 4–7)	Number of transmission retries on collision.
DEFAULT: F2h	



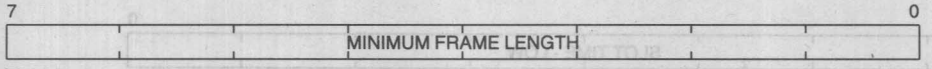
BYTE 8

PRM (Bit 0)	Promiscuous mode.
BC DIS (Bit 1)	Broadcast disable.
MANCH/NRZ (Bit 2)	Manchester or NRZ encoding. See specific timing requirements for TxC in Manchester mode.
TONO CRS (Bit 3)	Transmit on no CRS.
NOCRC INS (Bit 4)	No CRC insertion.
CRC-16/CRC-32 (Bit 5)	CRC type.
BIT STF (Bit 6)	Bit stuffing.
PAD (Bit 7)	Padding.
DEFAULT: 00h	



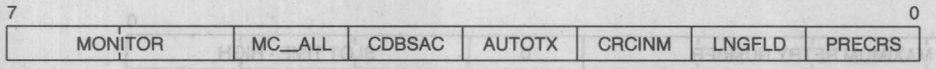
BYTE 9

CRSF (Bits 0–2)	Carrier Sense filter (length).
CRS SRC (Bit 3)	Carrier Sense source.
CDTF (Bits 4–6)	Collision Detect filter (length).
CDT SRC (Bit 7)	Collision Detect source.
DEFAULT: 00h	



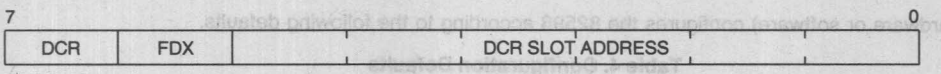
BYTE 10

MIN FRAME LEN	Minimum frame length.
DEFAULT: 40h	



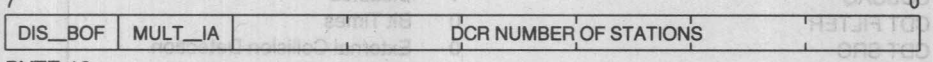
BYTE 11

PRECRS (Bit 0)	Preamble until Carrier Sense
LNGFLD (Bit 1)	Length field. Enables padding at the End-of-Carrier framing (802.3).
CRCINM (Bit 2)	Rx CRC appended to the frame in memory.
AUTOTX (Bit 3)	Auto retransmit.
CDBSAC (Bit 4)	Collision Detect by source address recognition.
MC__ALL (Bit 5)	Enable to receive all MC frames.
MONITOR (Bits 6–7)	Receive monitor options.
DEFAULT: FFH	



BYTE 12

DCR SLOT ADDRESS (Bits 0–5)	Station index in DCR mode.
FDX (Bit 6)	Enables Full Duplex operation.
DCR (Bit 7)	Enables Deterministic collision resolution.
DEFAULT: 00h	



BYTE 13

DCR NUMBER OF STATIONS (Bits 0–5)	Number of stations in DCR mode.
MULT_IA (Bit 6)	Multiple individual address.
DIS_BOF (Bit 7)	Disable the backoff algorithm.
DEFAULT: 3Fh	

1

A reset (hardware or software) configures the 82596 according to the following defaults.

Table 4. Configuration Defaults

	Parameter	Default Value	Units/Meaning
	ADDRESS LENGTH	**6	Bytes
	A/L FIELD LOCATION	0	Located in FD
*	AUTO RETRANSMIT	1	Auto Retransmit Enable
	BITSTUFFING/EOC	0	EOC
	BROADCAST DISABLE	0	Broadcast Reception Enabled
*	CDBSAC	1	Disabled
	CDT FILTER	0	Bit Times
	CDT SRC	0	External Collision Detection
*	CRC IN MEMORY	1	CRC Not Transferred to Memory
	CRC-16/CRC-32	**0	CRC-32
	CRS FILTER	0	0 Bit Times
	CRS SRC	0	External CRS
*	DCR	0	Disable DCR Protocol
*	DCR Slot Number	0	DCR Disabled
*	DCR Number of Stations	63	Stations
*	DISBOF	0	Backoff Enabled
	EXT LOOPBACK	0	Disabled
	EXPONENTIAL PRIORITY	**0	802.3 Algorithm
	EXPONENTIAL BACKOFF METHOD	**0	802.3 Algorithm
*	FULL DUPLEX (FDX)	0	CSMA/CD Protocol (No FDX)
	FIFO THRESHOLD	8	TX: 32 Bytes, RX: 64 Bytes
	INT LOOPBACK	0	Disabled
	INTERFRAME SPACING	**96	Bit Times
	LINEAR PRIORITY	**0	802.3 Algorithm
*	LENGTH FIELD	1	Padding Disabled
	MIN FRAME LENGTH	**64	Bytes
*	MC ALL	1	Disabled
*	MONITOR	11	Disabled
	MANCHESTER/NRZ	0	NRZ
*	MULTI IA	0	Disabled
	NUMBER OF RETRIES	**15	Maximum Number of Retries
	NO CRC INSERTION	0	CRC Appended to Frame
	PREFETCH BIT IN RBD	0	Disabled (Valid Only in New Modes)
	PREAMBLE LENGTH	**7	Bytes
*	Preamble Until CRS	1	Disabled
	PROMISCUOUS MODE	0	Address Filter On
	PADDING	0	No Padding
	SLOT TIME	**512	Bit Times
	SAVE BAD FRAME	0	Discards Bad Frames
	TRANSMIT ON NO CRS	0	Disabled

NOTES:
1. This configuration setup is compatible with the IEEE 802.3 specification.
2. The Asterisk "*" signifies a new configuration parameter not available in the 82586.
3. The default value of the Auto retransmit configuration parameter is enabled (1).
4. Double Asterisk "**" signifies IEEE 802.3 requirements.

MULTICAST-SETUP

This command is used to load the 82596 with the Multicast-IDs that should be accepted. As noted previously, the filtering done on the Multicast-IDs is not perfect and some unwanted frames may be accepted. This command resets the current filter and reloads it with the specified Multicast-IDs. The format of the Multicast-addresses setup command is:

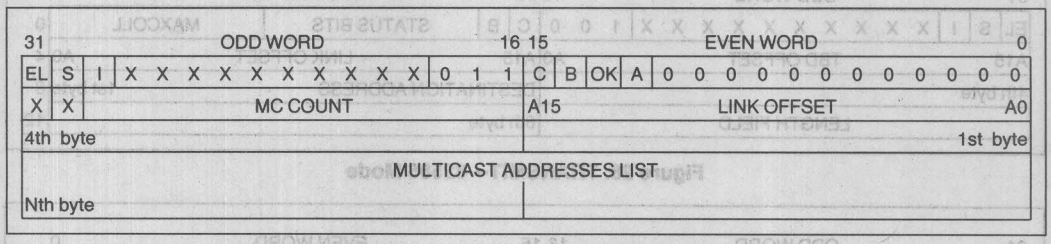


Figure 26. MC Setup—82586 and 32-Bit Segmented Modes

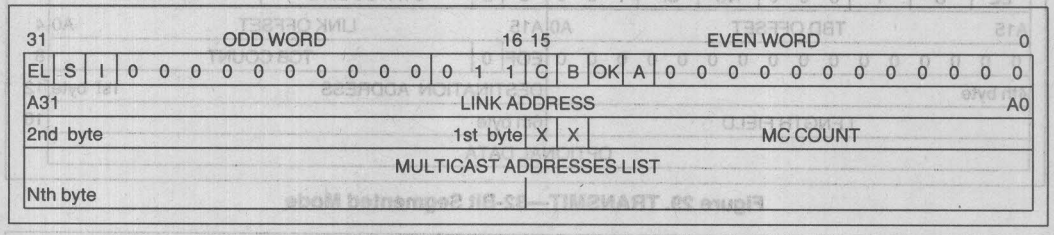


Figure 27. MC Setup—Linear Mode

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details)

A — Indicates that the command was abnormally terminated due to a CU Abort control command. If one, then the command was aborted and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in both the 32-Bit Segmented and Linear Modes).

CMD (bits 16–18) — The MC SETUP command value: 3h.

MC-CNT — This 14-bit field indicates the number of bytes in the MC LIST field. The MC CNT must be a multiple of the ADDR LEN; otherwise, the 82596 reduces the MC CNT to the nearest ADDR LEN multiple. MC CNT = 0 implies resetting the Hash table which is equivalent to disabling the Multicast filtering mechanism.

MC LIST — A list of Multicast Addresses to be accepted by the 82596. The least significant bit of each MC address must be 1.

NOTE:

The list is sequential; i.e., the most significant byte of an address is immediately followed by the least significant byte of the next address.

— When the 82596 is configured to recognize multiple Individual Address (Multi-IA), the MC-Setup command is also used to set up the Hash table for the individual address.

The least significant bit in the first byte of each IA address must be 0.

TRANSMIT

This command is used to transmit a frame of user data onto the serial link. The format of a Transmit command is as follows.

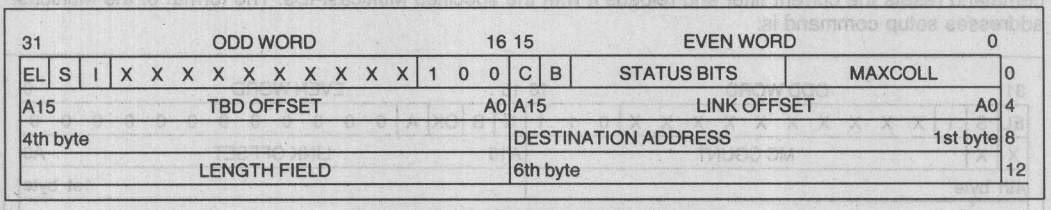


Figure 28. TRANSMIT—82586 Mode

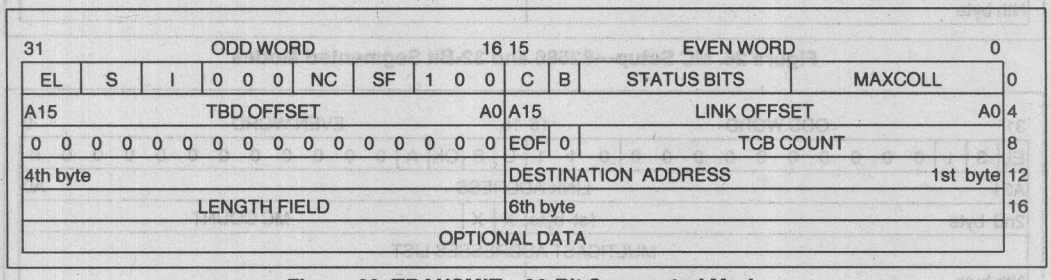


Figure 29. TRANSMIT—32-Bit Segmented Mode

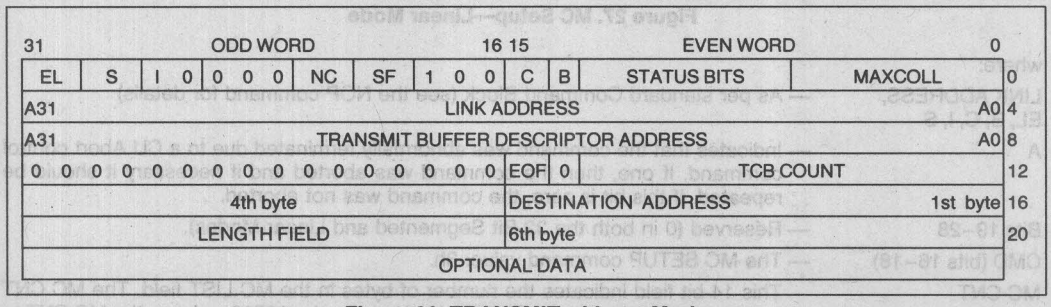
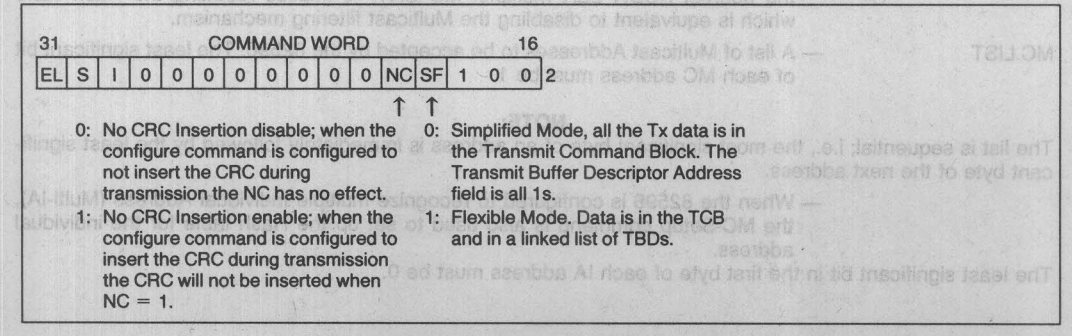


Figure 30. TRANSMIT—Linear Mode



where:

EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
OK (Bit 13)	— Error free completion.
A (Bit 12)	— Indicates that the command was abnormally terminated due to CU Abort control command. If 1, then the command was aborted, and if necessary it should be repeated. If this bit is 0, the command was not aborted.
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear modes).
CMD (Bits 16–18)	— The transmit command: 4h.
Status Bit 11	— Late collision. A late collision (a collision after the slot time is elapsed) is detected.
Status Bit 10	— No Carrier Sense signal during transmission. Carrier Sense signal is monitored from the end of Preamble transmission until the end of the Frame Check Sequence for TONOCRS = 1 (Transmit On No Carrier Sense mode) it indicates that transmission has been executed despite a lack of CRS. For TONOCRS = 0 (Ethernet mode), this bit also indicates unsuccessful transmission (transmission stopped when lack of Carrier Sense has been detected).
Status Bit 9	— Transmission unsuccessful (stopped) due to Loss of $\overline{\text{CTS}}$.
Status Bit 8	— Transmission unsuccessful (stopped) due to DMA Underrun; i.e., the system did not supply data for transmission.
Status Bit 7	— Transmission Deferred, i.e., transmission was not immediate due to previous link activity.
Status Bit 6	— Heartbeat Indicator, Indicates that after a previously performed transmission, and before the most recently performed transmission, (Interframe Spacing) the CDT signal was monitored as active. This indicates that the Ethernet Transceiver Collision Detect logic is performing properly. The Heartbeat is monitored during the Interframe Spacing period.
Status Bit 5	— Transmission attempt was stopped because the number of collisions exceeded the maximum allowable number of retries.
MAX-COL (Bits 3–0)	— The number of Collisions experienced during this frame. Max Col = 0 plus S5 = 1 indicates 16 collisions.
LINK OFFSET	— As per standard Command Block (see the NOP for details).
TBD POINTER	— In the 82586 and 32-bit Segmented modes this is the offset of the first Tx Buffer Descriptor containing the data to be transmitted. In the Linear mode this is the 32-bit address of the first Tx Buffer Descriptor on the list. If the TBD POINTER is all 1s it indicates that no TBD is used.
DEST ADDRESS	— Contains the Destination Address of the frame. The least significant bit (MC) indicates the address type. MC = 0: Individual Address. MC = 1: Multicast or Broadcast Address. If the Destination Address bits are all 1s this is a Broadcast Address.
LENGTH FIELD	— The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is transmitted; i.e., most significant byte first, least significant byte second.
TCB COUNT	— This 14-bit counter indicates the number of bytes that will be transmitted from the Transmit Command Block, starting from the third byte after the TCB COUNT field (address $n+12$ in the 32-bit Segmented mode, $N+16$ in the Linear mode). The TCB COUNT field can be any number of bytes (including an odd byte), this allows the user to transmit a frame with a header having an odd number of bytes. The TCB COUNT field is not used in the 82586 mode.
EOF Bit	— Indicates that the whole frame is kept in the Transmit Command Block. In the Simplified memory model it must be always asserted.

The interpretation of what is transmitted depends on the No Source Address insertion configuration bit and the memory model being used.

NOTES

1. The Destination Address and the Length Field are sequential of the Length Field immediately follows the most significant byte of the Destination Address.
2. In case the 82596 is configured with No Source Address insertion bit equal to 0, the 82596 inserts its configured Source Address in the transmitted frame.
   - In the 82586 mode, or when the Simplified memory model is used, the Destination and Length fields of the transmitted frame are taken from the Transmit Command Block.
   - If the FLEXIBLE memory model is used, the Destination and Length fields of the transmitted frame can be found either in the TCB or TBD, depending on the TCB COUNT.
3. If the 82596 is configured with the Address/Length Field Location equal to 1, the 82596 does not insert its configured Source Address in the transmitted frame. The first (2 × Address Length) + 2 bytes of the transmitted frame are interpreted as Destination Address, Source Address, and Length fields respectively. The location of the first transmitted byte depends on the operational mode of the 82596:
   - In the 82586 mode, it is always the first byte of the first Tx Buffer.
   - In both the 32-bit Segmented and Linear modes it depends on the SF bit and TCB COUNT:
     - — In the Simplified memory mode the first transmitted byte is always the third byte after the TCB COUNT field.
     - — In the Flexible mode, if the TCB COUNT is greater than 0 then it is the third byte after the TCB COUNT field. If TCB COUNT equals 0 then it is first byte of the first Tx Buffer.
   - Transmit frames shorter than six bytes are invalid. The transmission will be aborted (only in 82586 mode) because of a DMA Underrun.
4. Frames which are aborted during transmission are jammed. Such an interruption of transmission can be caused by any reason indicated by any of the status bits 8, 9, 10 and 12.

JAMMING RULES

1. Jamming will not start before completion of preamble transmission.
2. Collisions detected during transmission of the last 11 bits will not result in jamming.

The format of a Transmit Buffer Descriptor is:

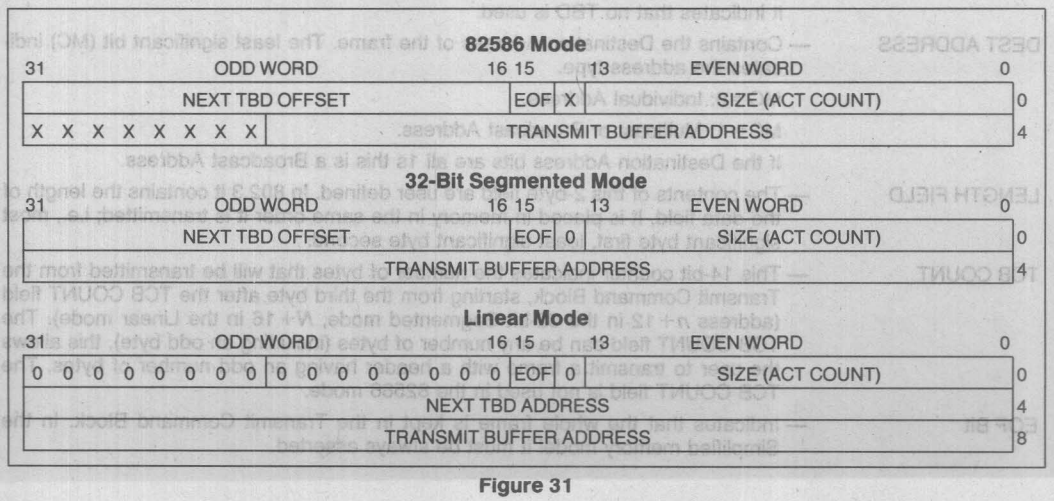


Figure 31

where:

EOF — This bit indicates that this TBD is the last one associated with the frame being transmitted. It is set by the CPU before transmit.

SIZE (ACT COUNT) — This 14-bit quantity specifies the number of bytes that hold information for the current buffer. It is set by the CPU before transmission.

NEXT TBD ADDRESS — In the 82586 and 32-bit Segmented modes, it is the offset of the next TBD on the list. In the Linear mode this is the 32-bit address of the next TBD on the list. It is meaningless if EOF = 1.

BUFFER ADDRESS — The starting address of the memory area that contains the data to be sent. In the 82586 mode, this is a 24-bit address (A31–A24 are considered to be zero). In the 32-bit Segmented and Linear modes this is a 32-bit address.

TDR

This operation activates Time Domain Reflectometry, which is a mechanism to detect open or short circuits on the link and their distance from the diagnosing station. The TDR command has no parameters. The TDR transmit sequence was changed, compared to the 82586, to form a regular transmission. The TDR bit stream is as follows.

— Preamble

— Source address

— Another Source address (the TDR frame is transmitted back to the sending station, so DEST ADR = SRC ADR).

— Data field containing 7Eh patterns.

— Jam Pattern, which is the inverse CRC of the transmitted frame.

Maximum length of the TDR frame is 2048 bits. If the 82596 senses collision while transmitting the TDR frame it transmits the jam pattern and stops the transmission. The 82596 then triggers an internal timer (STC); the timer is reset at the beginning of transmission and reset if CRS is returned. The timer measures the time elapsed from the start of transmission until an echo is returned. The echo is indicated by Collision Detect going active or a drop in the Carrier Sense signal. The following table lists the possible cases that the 82596 is able to analyze.

Conditions of TDR as Interpreted by the 82596

Transceiver Type / Condition	Ethernet	Non Ethernet
Carrier Sense was inactive for 2048-bit-time periods	Short or Open on the Transceiver Cable	NA
Carrier Sense signal dropped	Short on the Ethernet cable	NA
Collision Detect went active	Open on the Ethernet cable	Open on the Serial Link
The Carrier Sense Signal did not drop or the Collision Detect did not go active within 2048-bit time period	No Problem	No Problem

An Ethernet transceiver is defined as one that returns transmitted data on the receive pair and activates the Carrier Sense Signal while transmitting. A Non-Ethernet Transceiver is defined as one that does not do so.

The format of the Time Domain Reflectometer command is:

82586 and 32-Bit Segmented Modes

31 ODD WORD 16															15 EVEN WORD 0															
EL	S	I	X	X	X	X	X	X	X	X	X	X	1	0	1	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0
LNK OK	XVR PRB	ET OPN	ET SRT	X	TIME (11 bits)											A15	LINK OFFSET													A0

Linear Mode

31 ODD WORD 16															15 EVEN WORD 0															
EL	S	I	0	0	0	0	0	0	0	0	0	0	1	0	1	C	B	OK	0	0	0	0	0	0	0	0	0	0	0	0
A31	LINK ADDRESS																													A0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LNK OK	XVR PRB	ET OPN	ET SRT	X	TIME (11 bits)									

Figure 32. TDR

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).

A — Indicates that the command was abnormally terminated due to CU Abort control command. If one, then the command was aborted, and if necessary it should be repeated. If this bit is zero, the command was not aborted.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).

CMD (Bits 16–18) — The TDR command. Value: 5h.

TIME — An 11-bit field that specifies the number of TxC cycles that elapsed before an echo was observed. No echo is indicated by a reception consisting of "1s" only. Because the network contains various elements such as transceiver links, transceivers, Ethernet, repeaters etc., the TIME is not exactly proportional to the problems distance.

LNK OK (Bit 15) — No link problem identified. TIME = 7FFh.

XCVR PRB (Bit 14) — Indicates a Transceiver problem. Carrier Sense was inactive for 2048-bit time period. LNK OK = 0. TIME = 7FFh.

ET OPN (Bit 13) — The transmission line is not properly terminated. Collision Detect went active and LNK OK = 0.

ET SRT (Bit 12) — There is a short circuit on the transmission line. Carrier Sense Signal dropped and LNK OK = 0.

DUMP

This command causes the contents of various 82596 registers to be placed in a memory area specified by the user. It is supplied as a 82596 self-diagnostic tool, and to provide registers of interest to the user. The format of the DUMP command is:

82586 and 32-Bit Segmented Modes

31 ODD WORD 16	15 EVEN WORD 0
EL S I X X X X X X X X X X 1 1 0 C B OK	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A15 BUFFER OFFSET A0	A15 LINK OFFSET A0

Linear Mode

31 ODD WORD 16	15 EVEN WORD 0
EL S I X X X X X X X X X X 1 1 0 C B OK	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A31 LINK ADDRESS A0	
A31 BUFFER ADDRESS A0	

Figure 33. Dump

where:

LINK ADDRESS, EL, B, C, I, S — As per standard Command Block (see the NOP command for details).

OK — Indicates error free completion.

Bits 19–28 — Reserved (0 in the 32-bit Segmented and Linear Modes).

CMD (Bits 16–18) — The Dump command. Value: 6h.

BUFFER POINTER — In the 82586 and 32-bit Segmented modes this is the 16-bit-offset portion of the dump area address. In the Linear mode this is the 32-bit linear address of the dump area.

Dump Area Information Format

- The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
- In 82586 mode the dump area is 170 bytes.
- The dump area format of the 32-bit Segmented and Linear modes is described in Figure 35.
- The size of the dump area of the 32-bit Segmented and Linear modes is 304 bytes.
- When the dump is executed by the Port command an extra word will be appended to the Dump Area. The extra word is a copy of the Dump Area status word (containing the *C, B,* and *OK* bits). The *C* and *OK* bits are set when the 82596 has completed the Port Dump command.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
DMA CONTROL REGISTER*	00
CONFIGURE BYTES 3, 2	02
CONFIGURE BYTES 5, 4	04
CONFIGURE BYTES 7, 6	06
CONFIGURE BYTES 9, 8	08
CONFIGURE BYTES 10	0A
I.A. BYTES 1, 0*	0C
I.A. BYTES 3, 2*	0E
I.A. BYTES 5, 4*	10
LAST T.X. STATUS*	12
T.X. CRC BYTES 1, 0*	14
T.X. CRC BYTES 3, 2*	16
R.X. CRC BYTES 1, 0*	18
R.X. CRC BYTES 3, 2*	1A
R.X. TEMP MEMORY 1, 0*	1C
R.X. TEMP MEMORY 3, 2*	1E
R.X. TEMP MEMORY 5, 4*	20
LAST RECEIVED STATUS*	22
HASH REGISTER BYTES 1, 0*	24
HASH REGISTER BYTES 3, 2*	26
HASH REGISTER BYTES 5, 4*	28
HASH REGISTER BYTES 7, 6*	2A
SLOT TIME COUNTER*	2C
WAIT TIME COUNTER*	2E
MICRO MACHINE**	30
REGISTER FILE	.
60 BYTES	6A
MICRO MACHINE LFSR**	6C
MICRO MACHINE	6E
FLAG ARRAY	.
14 BYTES	7A
QUEUE MEMORY**	7C
CU PORT	.
8 BYTES	82
MICRO MACHINE ALU**	84
RESERVED**	86
M.M. TEMP A ROTATE R**	88
M.M. TEMP A**	8A
T.X. DMA BYTE COUNT**	8C
M.M. INPUT PORT ADDRESS**	8E
T.X. DMA ADDRESS**	90
M.M. OUTPUT PORT**	92
R.X. DMA BYTE COUNT**	94
R.U. DMA ADDRESS**	96
M.M. OUTPUT PORT ADDRESS REGISTER**	98
RESERVED**	9A
BUS THROTTLE TIMERS	9C
DIU CONTROL REGISTER**	9E
RESERVED**	A0
DMA CONTROL REGISTER**	A2
BIU CONTROL REGISTER**	A4
M.M. DISPATCHER REGISTER**	A6
M.M. STATUS REGISTER**	A8

NOTE:
*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 34. Dump Area Format—82586 Mode

31	0	
CONFIGURE BYTES 5, 4, 3, 2		00
CONFIGURE BYTES 9, 8, 7, 6		04
CONFIGURE BYTES 13, 12, 11, 10		08
I.A. BYTES 1, 0	X X X X X X X X	0C
I.A. BYTES 5, 2		10
TX CRC BYTES 0, 1	LAST T.X. STATUS	14
RX CRC BYTES 0, 1	TX CRC BYTES 3, 2	18
RX TEMP MEMORY 1, 0	RX CRC BYTES 3, 2	1C
R.X. TEMP MEMORY 5, 2		20
HASH REGISTERS 1, 0	LAST R.X. STATUS	24
HASH REGISTER BYTES 5, 2		28
SLOT TIME COUNTER	HASH REGISTERS 7, 6	2C
RECEIVE FRAME LENGTH	WAIT-TIME COUNTER	30
MICRO MACHINE**		34
REGISTER FILE 128 BYTES		. . B0
MICRO MACHINE LFSR**		B4
MICRO MACHINE**		B8
FLAG ARRAY 28 BYTES		. . D0
M.M. INPUT PORT**		D4
16 BYTES		E0
MICRO MACHINE ALU**		E4
RESERVED**		E8
M.M. TEMP A ROTATE R.**		EC
M.M. TEMP A**		F0
T.X. DMA BYTE COUNT**		F4
M.M. INPUT PORT ADDRESS REGISTER**		F8
T.X. DMA ADDRESS**		FC
M.M. OUTPUT PORT REGISTER**		100
R.X. DMA BYTE COUNT**		104
M.M. OUTPUT PORT ADDRESS REGISTER**		108
R.X. DMA ADDRESS REGISTER**		10C
RESERVED**		110
BUS THROTTLE TIMERS		114
DIU CONTROL REGISTER**		118
RESERVED**		11C
DMA CONTROL REGISTER**		120
BIU CONTROL REGISTER**		124
M.M. DISPATCHER REG.**		128
M.M. STATUS REGISTER**		12C

NOTE:
*The 82596 is not Dump compatible with the 82586 because of the 32-bit internal architecture. In 82586 mode the 82596 will dump the same number of bytes as the 82586. The compatible data will be marked with an asterisk.
**These bytes are not user defined, results may vary from Dump command to Dump command.

Figure 35. Dump Area Format—Linear and 32-Bit Segmented Mode

DIAGNOSE

The Diagnose Command triggers an internal self-test procedure that checks internal 82596 hardware, which includes:

- Exponential Backoff Random Number Generator (Linear Feedback Shift Register).
- Exponential Backoff Timeout Counter.
- Slot Time Period Counter.
- Collision Number Counter.
- Exponential Backoff Shift Register.
- Exponential Backoff Mask Logic.
- Timer Trigger Logic.

This procedure checks the operation of the Backoff block, which resides in the serial side and is not easily controlled. The Diagnose command is performed in two phases.

The format of the 82596 Diagnose command is:

82586 and 32-Bit Segmented Modes

31															16	15															0
ODD WORD																EVEN WORD															
EL	S	I	X	X	X	X	X	X	X	X	X	X	1	1	1	C	B	OK	0	F	0	0	0	0	0	0	0	0	0	0	0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	A15															A0
																LINK OFFSET															

Linear Mode

31															16	15															0
ODD WORD																EVEN WORD															
EL	S	I	0	0	0	0	0	0	0	0	0	0	1	1	1	C	B	OK	0	F	0	0	0	0	0	0	0	0	0	0	0
A31																LINK ADDRESS															A0

Figure 36. Diagnose

where:

LINK ADDRESS, EL, B, C, I, S	— As per standard Command Block (see the NOP command for details).
Bits 19–28	— Reserved (0 in the 32-bit Segmented and Linear Modes).
CMD (bits 16–18)	— The Diagnose command. Value: 7h.
OK (bit 13)	— Indicates error free completion.
F (bit 11)	— Indicates that the self-test procedure has failed.

RECEIVE FRAME DESCRIPTOR

Each received frame is described by one Receive Frame Descriptor (see Figure 37). Two new memory structures are available for the received frames. The structures are available only in the Linear and 32-bit Segmented modes.

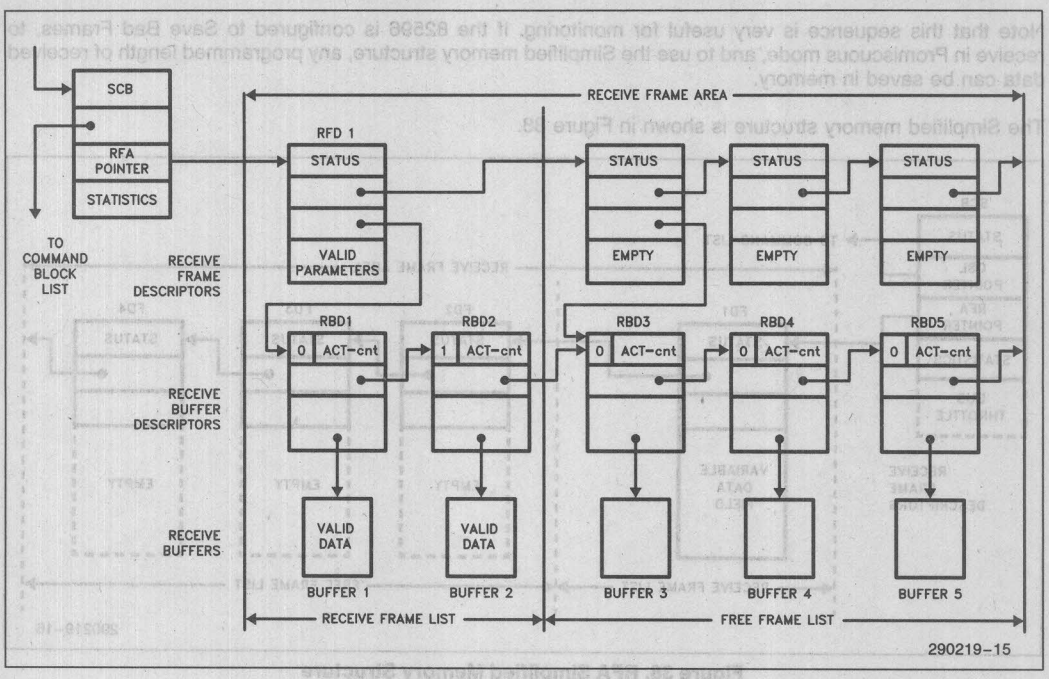


Figure 37. The Receive Frame Area

Simplified Memory Structure

The first is the Simplified memory structure, the data section of the received frame is part of the RFD and is located immediately after the Length Field. Receive Buffer Descriptors are not used with the Simplified structure, it is primarily used to make programming easier. If the length of the data area described in the Size Field is smaller than the incoming frame, the following happens.

1. The received frame is truncated.
2. The No Resource error counter is updated.
3. If the 82596 is configured to Save Bad Frames the RFD is not reused; otherwise, the same RFD is used to hold the next received frame, and the only action taken regarding the truncated frame is to update the counter.
4. The 82596 continues to receive the next frame in the next RFD.

Note that this sequence is very useful for monitoring. If the 82596 is configured to Save Bad Frames, to receive in Promiscuous mode, and to use the Simplified memory structure, any programmed length of received data can be saved in memory.

The Simplified memory structure is shown in Figure 38.

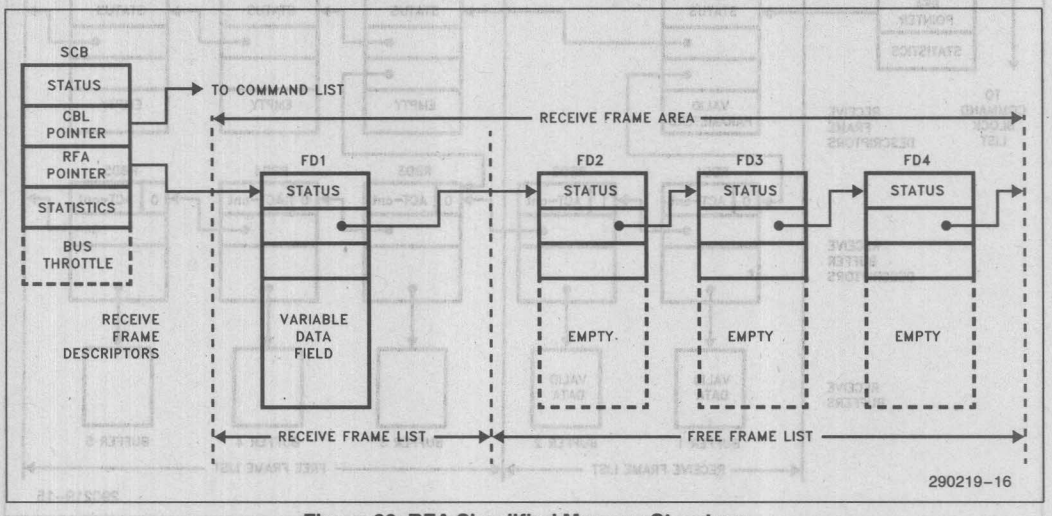


Figure 38. RFA Simplified Memory Structure

Flexible Memory Structure

The second structure is the Flexible memory structure, the data structure of the received frame is stored in both the RFD and in a linked list of Receive Buffers—Receive Buffer Descriptors. The received frame is placed in the RFD as configured in the Size field. Any remaining data is placed in a linked list of RBDs.

The Flexible memory structure is shown in Figure 39.

Buffers on the receive side can be different lengths. The 82596 will not place more bytes into a buffer than indicated in the associated RBD. The 82596 will fetch the next RBD before it is needed. The 82596 will attempt to receive frames as long as the FBL is not exhausted. If there are no more buffers, the 82596 Receive Unit will enter the No Resources state. Before starting the RU, the CPU must place the FBL pointer in the RBD pointer field of the first RFD. All remaining RBD pointer fields for subsequent RFDs should be "1s." If the Receive Frame Descriptor and the associated Receive Buffers are not reused (e.g., the frame is properly received or the 82596 is configured to Save Bad Frames), the 82596 writes the address of the next free RBD to the RBD pointer field of the next RFD.

RECEIVE BUFFER DESCRIPTOR (RBD)

The RBDs are used to store received data in a flexible set of linked buffers. The portion of the frame's data field that is outside the RFD is placed in a set of buffers chained by a sequence of RBDs. The RFD points to the first RBD, and the last RBD is flagged with an EOF bit set to 1. Each buffer in the linked list of buffers related to a particular frame can be any size up to 2^{14} bytes but must be word aligned (begin on an even numbered byte). This ensures optimum use of the memory resources while maintaining low overhead. All buffers in a frame are filled with the received data except for the last, in which the actual count can be smaller than the allocated buffer space.

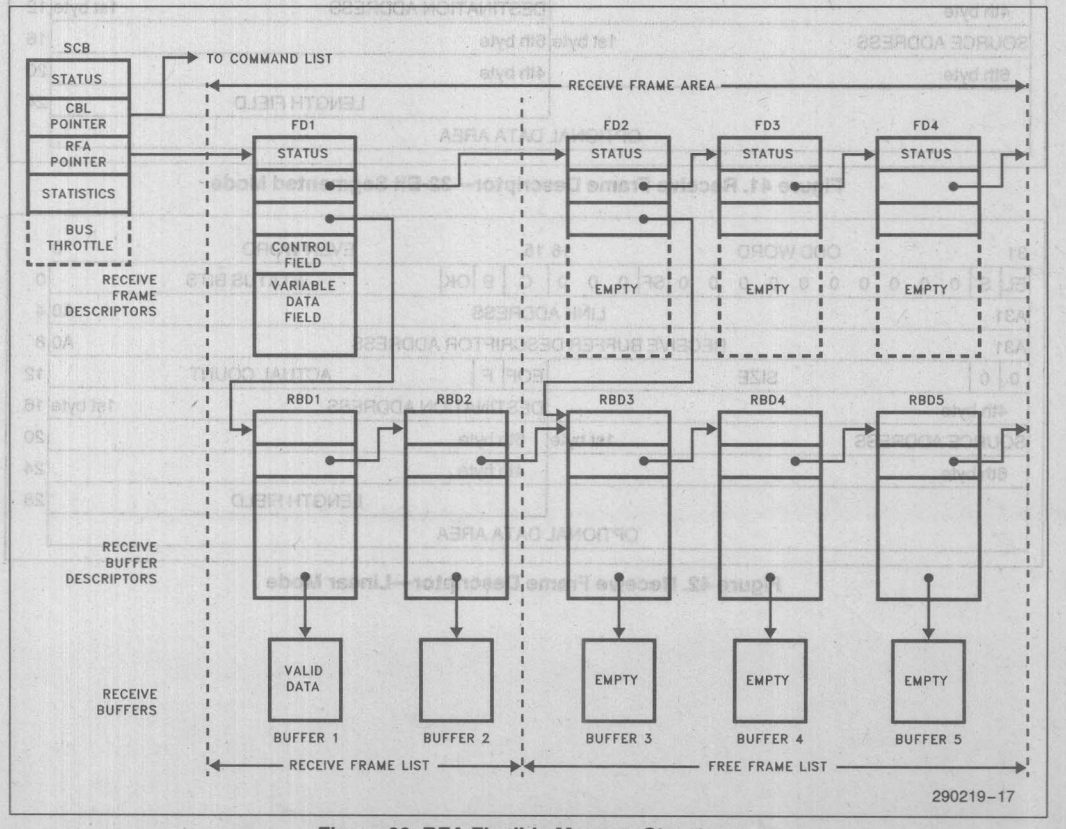


Figure 39. RFA Flexible Memory Structure

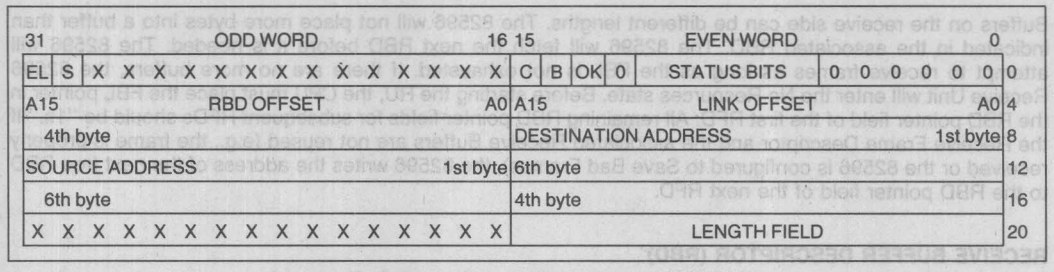


Figure 40. Receive Frame Descriptor—82586 Mode

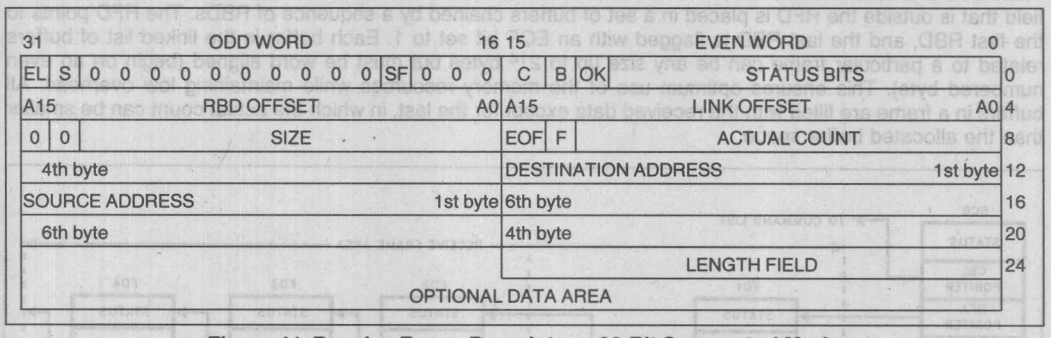


Figure 41. Receive Frame Descriptor—32-Bit Segmented Mode

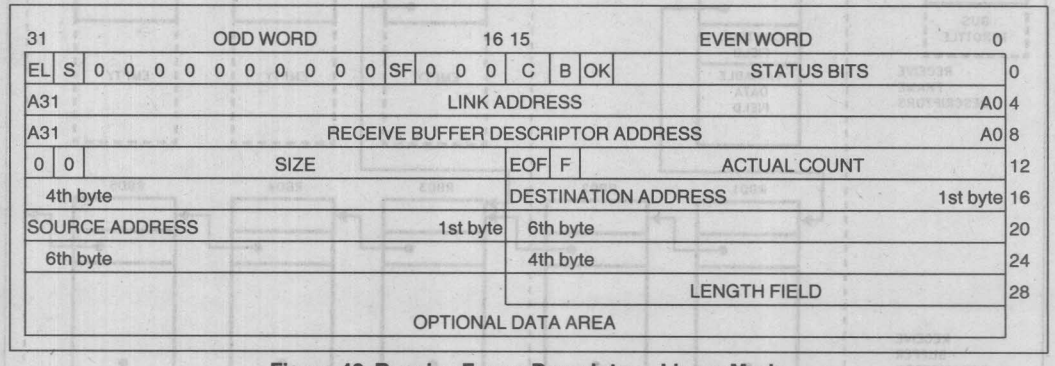


Figure 42. Receive Frame Descriptor—Linear Mode

where:

EL — When set, this bit indicates that this RFD is the last one on the RDL.

S — When set, this bit suspends the RU after receiving the frame.

SF — This bit selects between the Simplified or the Flexible mode.

0 — Simplified mode, all the RX data is in the RFD. RBD ADDRESS field is all "1s."

1 — Flexible mode. Data is in the RFD and in a linked list of Receive Buffer Descriptors.

C — This bit indicates the completion of frame reception. It is set by the 82596.

B — This bit indicates that the 82596 is currently receiving this frame, or that the 82596 is ready to receive the frame. It is initially set to 0 by the CPU. The 82596 sets it to 1 when reception set up begins, and to 0 upon completion. The C and B bits are set during the same operation.

OK (bit 13) — Frame received successfully, without errors. RFDs with bit 13 equal to 0 are possible only if the save bad frames configuration option is selected. Otherwise all frames with errors will be discarded, although statistics will be collected on them.

STATUS — The results of the Receive operation. Defined bits are,

Bit 12: Length error if configured to check length

Bit 11: CRC error in an aligned frame

Bit 10: Alignment error (CRC error in misaligned frame)

Bit 9: Ran out of buffer space—no resources

Bit 8: DMA Overrun failure to acquire the system bus.

Bit 7: Frame too short.

Bit 6: No EOP flag (for Bit stuffing only)

Bit 5: When the SF bit equals zero, and the 82596 is configured to save bad frames, this bit signals that the received frame was truncated. Otherwise it is zero.

Bits 2–4: Zeros

Bit 1: When it is zero, the destination address of the received frame matches the IA address. When it is 1, the destination address of the received frame does not match the individual address. For example, a multicast address or broadcast address will set this bit to a 1.

Bit 0: Receive collision, a collision is detected during reception.

LINK ADDRESS — A 16-bit offset (32-bit address in the Linear mode) to the next Receive Frame Descriptor. The Link Address of the last frame can be used to form a cyclical list.

RBD POINTER — The offset (address in the Linear mode) of the first RBD containing the received frame data. An RBD pointer of all ones indicates no RBD.

EOF
F
SIZE
ACT COUNT — These fields are for the Simplified and Flexible memory models. They are exactly the same as the respective fields in the Receive Buffer Descriptor. See the next section for detailed explanation of their functions.

MC — Multicast bit.

DESTINATION ADDRESS — The contents of the destination address of the receive frame. The field is 0 to 6 bytes long.

SOURCE ADDRESS — The contents of the Source Address field of the received frame. It is 0 to 6 bytes long.

LENGTH FIELD — The contents of this 2-byte field are user defined. In 802.3 it contains the length of the data field. It is placed in memory in the same order it is received, i.e., most significant byte first, least significant byte second.

NOTES

1. The Destination address, Source address and Length fields are packed, i.e., one field immediately follows the next.
2. The affect of Address/Length Location (No Source Address Insertion) configuration parameter while receiving is as follows:

— 82586 Mode: The Destination address, Source address and Length field are not used, they are placed in the RX data buffers.

— 32-Bit Segmented and Linear Modes: when the Simplified memory model is used, the Destination address, Source address and Length fields reside in their respective fields in the RFD. When the Flexible memory strucrture is used the Destination address, Source address, and Length field locations depend on the SIZE field of the RFD. They can be placed in the RFD, in the RX data buffers, or partially in the RFD and the rest in the RX data buffers, depending on the SIZE field value.

82586 Mode

31 ODD WORD 16	15 EVEN WORD 0	
A15 NEXT RBD OFFSET A0	EOF F ACTUAL COUNT	0
X X X X X X X X A23 RECEIVE BUFFER ADDRESS A0		4
X X X X X X X X X X X X X X X X	EL X SIZE	8

32-Bit Segmented Mode

31 ODD WORD 16	15 EVEN WORD 0	
A15 NEXT RBD OFFSET A0	EOF F ACTUAL COUNT	0
A31 RECEIVE BUFFER ADDRESS A0		4
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	EL P SIZE	8

Linear Mode

31 ODD WORD 16	15 EVEN WORD 0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	EOF F ACTUAL COUNT	0
A31 NEXT RBD ADDRESS A0		4
A31 RECEIVE BUFFER ADDRESS A0		8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	EL P SIZE	

Figure 43. Receive Buffer Descriptor

where:

EOF — Indicates that this is the last buffer related to the frame. It is cleared by the CPU before starting the RU, and is written by the 82596 at the end of reception of the frame.

F — Indicates that this buffer has already been used. The Actual Count has no meaning unless the F bit equals one. This bit is cleared by the CPU before starting the RU, and is set by the 82596 after the associated buffer has been. This bit has the same meaning as the Complete bit in the RFD and CB.

ACT COUNT — This 14-bit quantity indicates the number of meaningful bytes in the buffer. It is cleared by the CPU before starting the RU, and is written by the 82596 after the associated buffer has already been used. In general, after the buffer is full, the Actual Count value equals the size field of the same buffer. For the last buffer of the frame, Actual Count can be less than the buffer size.

NEXT BD ADDRESS — The offset (absolute address in the Linear mode) of the next RBD on the list. It is meaningless if EL=1.

BUFFER ADDRESS — The starting address of the memory area that contains the received data. In the 82586 mode, this is a 24-bit address (with pins A24–A31=0). In the 32-bit Segmented and Linear modes this is a 32-bit address.

EL — Indicates that the buffer associated with this RBD is last in the FBL.

P — This bit indicates that the 82596 has already prefetched the RBDs and any change in the RBD data will be ignored. This bit is valid only in the new 82596 memory modes, and if this feature has been enabled during configure command. The 82596 Prefetches the RBDs in locked cycles; after prefetching the RBD the 82596 performs a write cycle where the P bit is set to one and the rest of the data remains unchanged. The CPU is responsible for resetting it in all RBDs. The 82596 will not check this bit before setting it.

SIZE — This 14-bit quantity indicates the size, in bytes, of the associated buffer. This quantity must be an even number.

ELECTRICAL AND TIMING CHARACTERISTICS

D.C. CHARACTERISTICS

T_C = 0°C–85°C, V_{CC} = 5V ±10% CLK2 and LE/$\overline{BE}$ have MOS levels (see V_{MIL}, V_{MIH}).
All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, V_{OH}).

Symbol	Parameter	Min	Max	Units	Notes
V_{IL}	Input Low Voltage (TTL)	−0.3	+0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.3	V	
V_{MIL}	Input Low Voltage (MOS)	−0.3	+0.8	V	
V_{MIH}	Input High Voltage (MOS)	3.7	V_{CC} + 0.3	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 4.0 mA(1)
V_{CIL}	$\overline{RxC}$, $\overline{TxC}$ Input Low Voltage	−0.5	0.6	V	
V_{CIH}	$\overline{RxC}$, $\overline{TxC}$ Input High Voltage	3.3	V_{CC} + 0.5	V	
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} = 0.9mA–1 mA(1)
I_{LI}	Input Leakage Current		±15	μA	0 ≤ V_{IN} ≤ V_{CC}
I_{LO}	Output Leakage Current		±15	μA	0.45 < V_{OUT} < V_{CC}
C_{IN}	Capacitance of Input Buffer		10	pF	FC = 1 MHz
C_{OUT}	Capacitance of Input/Output Buffer		12	pF	FC = 1 MHz
C_{CLK}	CLK Capacitance		20	pF	FC = 1 MHz
I_{CC}	Power Supply		150	mA	At 16 MHz for the 82596SX
I_{CC}	Power Supply		200	mA	At 25 MHz
I_{CC}	Power Supply		300	mA	At 33 MHz

SEE INTEL PRELIMINARY DATASHEET FOR LATEST SUPPLEMENT

A.C. CHARACTERISTICS

82596DX INPUT/OUTPUT SYSTEM TIMINGS $T_C = 0°C$ to $+85°$, $V_{CC} = 5V \pm 10\%$

These timings assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however, timings must be derated.
All timing requirements are given in nanoseconds.

Symbol	Parameter	25 MHz		Notes
		Min	Max	
	Operating Frequency	12 MHz	25 MHz	CLK2/2
T1	CLK2 Period	20	40	
T2	CLK2 High	4		3.7V
T3	CLK2 Low	5		0.8V
T4	CLK2 Rise Time	—	7	0.8V to 3.7V
T5	CLK2 Fall Time	—	7	3.7V to 0.8V
T13	CA and BREQ Setup Time	7		1, 2, 3
T14	CA and BREQ Hold Time	3		1, 2, 3
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	4 T1		3
T25	INT Valid Delay	1	26	
T6	$\overline{BEx}$, $\overline{LOCK}$, and A2–A31 Valid Delay	4	21	
T7	$\overline{BEx}$, $\overline{LOCK}$, and A2–A31 Float Delay	4	30	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	4	21	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	4	30	
T10	D0–D31 Write Data Valid Delay	3	22	
T11	D0–D31 Write Data Float Delay	4	22	
T27	D0–D31 CPU $\overline{PORT}$ Access Setup Time	7		2
T28	D0–D31 CPU $\overline{PORT}$ Access Hold Time	5		2
T29	$\overline{PORT}$ Setup Time	7		2
T30	$\overline{PORT}$ Hold Time	3		2
T17	$\overline{RDY}$ Setup Time	9		2
T18	$\overline{RDY}$ Hold Time	3		2
T19	D0–D31 READ Setup Time	7		2
T20	D0–D31 READ Hold Time	5		2
T12	HOLD Valid Delay	4	22	
T21	HLDA Setup Time	10		1, 2
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	10		2
T24	RESET Hold Time	3		2

PRELIMINARY
SEE INTEL FOR DATASHEET
SUPPLEMENT

A.C. CHARACTERISTICS (Continued)

82596DX INPUT/OUTPUT SYSTEM TIMINGS $T_C = 0°C$ to $+85°C$, $V_{CC} = 5V \pm 5\%$

These timings assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however, timings must be derated.
All timing requirements are given in nanoseconds.

Symbol	Parameter	33 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	33 MHz	CLK2/2
T1	CLK2 Period	15	40	
T2	CLK2 High	4.5		3.7V
T3	CLK2 Low	4.5		0.8V
T4	CLK2 Rise Time	—	4	3.7V to 0.8V
T5	CLK2 Fall Time	—	4	0.8V to 3.7V
T13	CA and BREQ Setup Time	7		1, 2, 3
T14	CA and BREQ Hold Time	3		1, 2, 3
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	4 T1		3
T25	INT Valid Delay	1	20	
T6	$\overline{BEx}$, $\overline{LOCK}$, and A2–A31 Valid Delay	4	15	
T7	$\overline{BEx}$, $\overline{LOCK}$, and A2–A31 Float Delay	4	20	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	4	15	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	4	20	
T10	D0–D31 Write Data Valid Delay	3	19	
T11	D0–D31 Write Data Float Delay	4	17	
T27	D0–D31 CPU $\overline{PORT}$ Access Setup Time	5		2
T28	D0–D31 CPU $\overline{PORT}$ Access Hold Time	3		2
T29	$\overline{PORT}$ Setup Time	7		2
T30	$\overline{PORT}$ Hold Time	3		2
T17	$\overline{RDY}$ Setup Time	7		2
T18	$\overline{RDY}$ Hold Time	3		2
T19	D0–D31 READ Setup Time	5		2
T20	D0–D31 READ Hold Time	3		2
T12	HOLD Valid Delay	3	19	
T21	HLDA Setup Time	8		1, 2
T22a	HLDA Hold Time	3		1, 2
T23	RESET Setup Time	8		2
T24	RESET Hold Time	3		2

PRELIMINARY
SEE INTERIM FOR DATASHEET
SUPPLEMENT

A.C. CHARACTERISTICS (Continued)

82596SX INPUT/OUTPUT SYSTEM TIMINGS T_C = 0°C to +85°C, V_{CC} = 5V ±10%

These timings assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however, timings must be derated.
All timing requirements are given in nanoseconds.

Symbol	Parameter	20 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	20 MHz	CLK2/2
T1	CLK2 Period	25	40	
T2	CLK2 High	8		at 2.0V
T3	CLK2 Low	8		at 2.0V
T4	CLK2 Rise Time	—	8	0.8V to 3.7V
T5	CLK2 Fall Time	—	8	3.7V to 0.8V
T13	CA and BREQ Setup Time	10		1, 2, 3
T14	CA and BREQ Hold Time	8		1, 2, 3
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	4 T1		3
T25	INT Valid Delay	1	35	
T6	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$, BON, and A1–A31 Valid Delay	4	30	
T7	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$, BON, and A1–A31 Float Delay	4	30	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	4	26	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	4	30	
T10	D0–D15 Write Data Valid Delay	4	38	
T11	D0–D15 Write Data Float Delay	4	27	
T27	D0–D15 CPU $\overline{PORT}$ Access Setup Time	9		2
T28	D0–D15 CPU $\overline{PORT}$ Access Hold Time	6		2
T29	$\overline{PORT}$ Setup Time	10		2
T30	$\overline{PORT}$ Hold Time	8		2
T17	$\overline{RDY}$ Setup Time	12		2
T18	$\overline{RDY}$ Hold Time	5.5		2
T19	D0–D15 READ Setup Time	9		2
T20	D0–D15 READ Hold Time	6		2
T12	HOLD Valid Delay	4	28	
T21	HLDA Setup Time	15		1, 2
T22a	HLDA Hold Time	8		1, 2
T23	RESET Setup Time	12		1, 2
T24	RESET Hold Time	4		1, 2

PRELIMINARY DATASHEET SUPPLEMENT FOR INTEL USE ONLY

A.C. CHARACTERISTICS (Continued)

82596SX INPUT/OUTPUT SYSTEM TIMINGS $T_C = 0°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$

These timings assume the C_L on all outputs is 50 pF unless otherwise specified. C_L can be 20 pF to 120 pF, however, timings must be derated.
All timing requirements are given in nanoseconds.

Symbol	Parameter	16 MHz		Notes
		Min	Max	
	Operating Frequency	12.5 MHz	16 MHz	CLK2/2
T1	CLK2 Period	31	40	
T2	CLK2 High	9		2.0V
T3	CLK2 Low	9		2.0V
T4	CLK2 Rise Time	—	8	0.8V to 3.7V
T5	CLK2 Fall Time	—	8	3.7V to 0.8V
T13	CA and BREQ Setup Time	11		1, 2, 3
T14	CA and BREQ Hold Time	8		1, 2, 3
T26	CA and BREQ, $\overline{PORT}$ Pulse Width	4 T1		3
T25	INT Valid Delay	1	40	
T6	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$, BON, and A1–A31 Valid Delay	4	36	
T7	$\overline{BHE}$, $\overline{BLE}$, $\overline{LOCK}$, BON, and A1–A31 Float Delay	4	40	
T8	W/$\overline{R}$ and $\overline{ADS}$ Valid Delay	4	33	
T9	W/$\overline{R}$ and $\overline{ADS}$ Float Delay	4	35	
T10	D0–D15 Write Data Valid Delay	4	40	
T11	D0–D15 Write Data Float Delay	4	35	
T27	D0–D15 CPU $\overline{PORT}$ Access Setup Time	9		2
T28	D0–D15 CPU $\overline{PORT}$ Access Hold Time	6		2
T29	$\overline{PORT}$ Setup Time	11		2
T30	$\overline{PORT}$ Hold Time	8		2
T17	$\overline{RDY}$ Setup Time	19		2
T18	$\overline{RDY}$ Hold Time	6		2
T19	D0–D15 READ Setup Time	9		2
T20	D0–D15 READ Hold Time	6		2
T12	HOLD Valid Delay	4	33	
T21	HLDA Setup Time	15		1, 2
T22a	HLDA Hold Time	8		1, 2
T23	RESET Setup Time	13		1, 2
T24	RESET Hold Time	4		1, 2

PRELIMINARY INTEL DATASHEET SUPPLEMENT

NOTES:
1. RESET, HLDA, and CA are internally synchronized. This timing is to guarantee recognition at next clock for RESET, HLDA, and CA.
2. All set-up, hold, and delay timings are at the maximum frequency specification Fmax, and must be derated according to the following equation for operation at lower frequencies:
   Tderated = (Fmax/Fopr) × T
   where:
   Tderated = Specifies the value to derate the specification.
   Fmax = Maximum operating frequency.
   Fopr = Actual operating frequency.
   T = Specification at maximum frequency.
   This calculation only provides a rough estimate for derating the frequency. For more detailed information contact your Intel sales office for the data sheet supplement.
3. CA is internally synchronized; if the setup and hold times are met then CA needs to be only 2 T1. BREQ and $\overline{PORT}$ are not internally synchronized. BREQ must meet setup and hold times and need only be 2 T1 wide.

TRANSMIT/RECEIVE CLOCK PARAMETERS

Symbol	Parameter	20 MHz		Notes
		Min	Max	
T36	$\overline{TxC}$ Cycle	50		1, 3
T38	$\overline{TxC}$ Rise Time		5	1
T39	$\overline{TxC}$ Fall Time		5	1
T40	$\overline{TxC}$ High Time	19		1, 3
T41	$\overline{TxC}$ Low Time	18		1, 3
T42	TxD Rise Time		10	4
T43	TxD Fall Time		10	4
T44	TxD Transition	20		2, 4
T45	$\overline{TxC}$ Low to TxD Valid		25	4, 6
T46	$\overline{TxC}$ Low to TxD Transition		25	2, 4
T47	$\overline{TxC}$ High to TxD Transition		25	2, 4
T48	$\overline{TxC}$ Low to TxD High (At End of Transition)		25	4
$\overline{RTS}$ AND $\overline{CTS}$ PARAMETERS				
T49	$\overline{TxC}$ Low to $\overline{RTS}$ Low, Time to Activate $\overline{RTS}$		25	5
T50	$\overline{CTS}$ Low to $\overline{TxC}$ Low, $\overline{CTS}$ Setup Time		20	
T51	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid, $\overline{CTS}$ Hold Time	10		7
T52	$\overline{TxC}$ Low to $\overline{RTS}$ High		25	5
RECEIVE CLOCK PARAMETERS				
T53	$\overline{RxC}$ Cycle	50		1, 3
T54	$\overline{RxC}$ Rise Time		5	1
T55	$\overline{RxC}$ Fall Time		5	1
T56	$\overline{RxC}$ High Time	19		1
T57	$\overline{RxC}$ Low Time	18		1
RECEIVED DATA PARAMETERS				
T58	RxD Setup Time	20		6
T59	RxD Hold Time	10		6

SEE INTEL PRELIMINARY DATASHEET FOR SUPPLEMENT

TRANSMIT/RECEIVE CLOCK PARAMETERS (Continued)

Symbol	Parameter	20 MHz		Notes
		Min	Max	
RECEIVED DATA PARAMETERS (Continued)				
T60	RxD Rise Time		10	
T61	RxD Fall Time		10	
$\overline{CRS}$ AND $\overline{CDT}$ PARAMETERS				
T62	$\overline{CDT}$ Low to $\overline{TxC}$ HIGH External Collision Detect Setup Time	20		
T63	$\overline{TxC}$ High to $\overline{CDT}$ Inactive, $\overline{CDT}$ Hold Time	10		
T64	$\overline{CDT}$ Low to Jam Start			10
T65	$\overline{CRS}$ Low to $\overline{TxC}$ High, Carrier Sense Setup Time	20		
T66	$\overline{TxC}$ High to $\overline{CRS}$ Inactive, $\overline{CRS}$ Hold Time	10		
T67	$\overline{CRS}$ High to Jamming Start, (Internal Collision Detect)			12
T68	Jamming Period			11
T69	$\overline{CRS}$ High to $\overline{RxC}$ High, $\overline{CRS}$ Inactive Setup Time	30		
T70	$\overline{RxC}$ High to $\overline{CRS}$ High, $\overline{CRS}$ Inactive Hold Time	10		
INTERFRAME SPACING PARAMETERS				
T71	Interframe Delay			9
EXTERNAL LOOPBACK-PIN PARAMETERS				
T72	$\overline{TxC}$ Low to $\overline{LPBK}$ Low		T36	4
T73	$\overline{TxC}$ Low to $\overline{LPBK}$ High		T36	4

SEE INTEL PRELIMINARY DATASHEET FOR SUPPLEMENT

NOTES:
1. Special MOS levels, V_{CIL} = 0.9V and V_{CIH} = 3.0V.
2. Manchester only.
3. Manchester. Needs 50% duty cycle.
4. 1 TTL load + 50 pF.
5. 1 TTL load + 100 pF.
6. NRZ only.
7. Abnormal end of transmission—CTS expires before RTS.
8. Normal end to transmission.
9. Programmable value:
   T71 = N_{IFS} • T36
   where: N_{IFS} = the IFS configuration value
   (if N_{IFS} is less than 12 then N_{IFS} is forced to 12).
10. Programmable value:
    T64 = (N_{CDF} • T36) + x • T36
    (If the collision occurs after the preamble)
    where:
    N_{CDF} = the collision detect filter configuration value, and
    x = 12, 13, 14, or 15
11. T68 = 32 • T36
12. Programmable value:
    T67 = (N_{CSF} • T36) + x • T36
    where: N_{CSF} = the Carrier Sense Filter configuration value, and
    x = 12, 13, 14, or 15
13. To guarantee recognition on the next clock.

82596DX/SX BUS OPERATION

The following figures show thae basic bus cycles for the 82596DX and 82596SX.

For more details refer to the *32-Bit LAN Components Manual.*

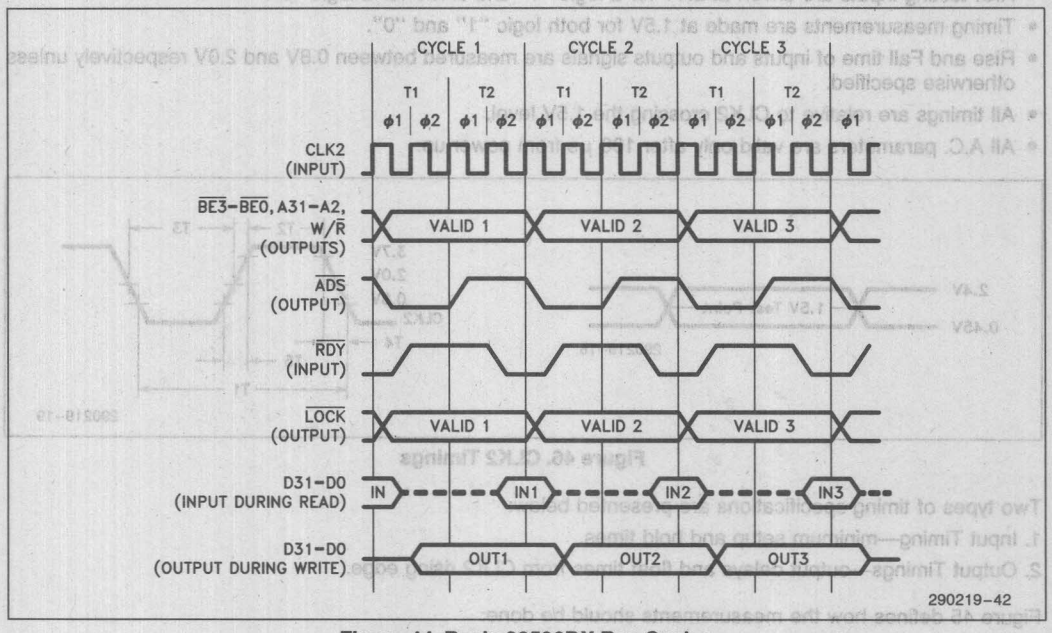


Figure 44. Basic 82596DX Bus Cycles

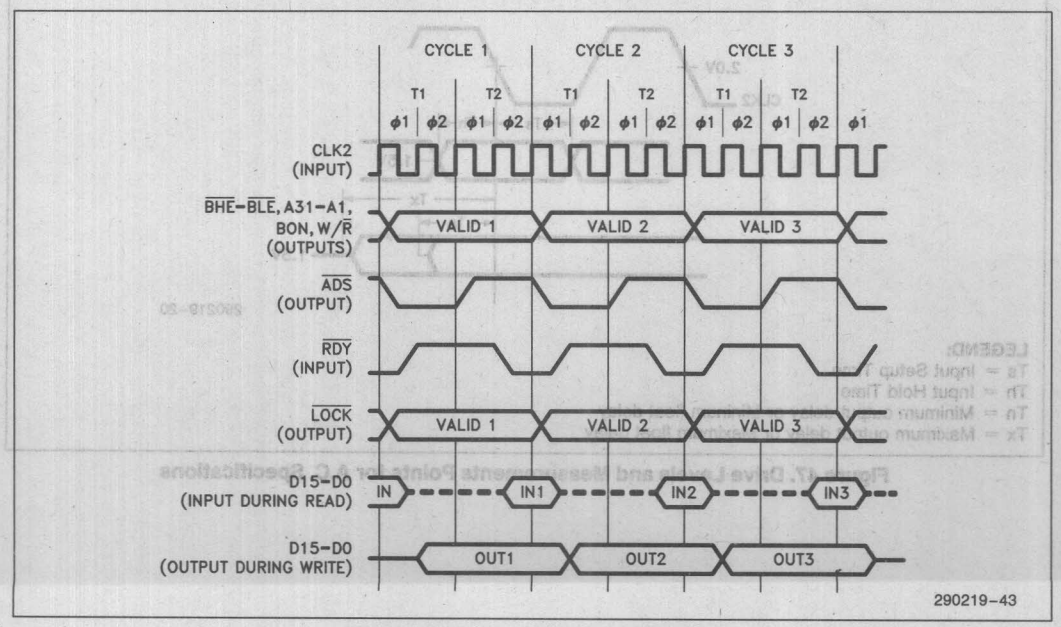


Figure 45. Basic 82596SX Bus Cycles

SYSTEM INTERFACE A.C. TIMING CHARACTERISTICS

The measurements should be done at:

- T_C = 0°C–85°C, V_{CC} = 5V ±10%, C = 50 pF unless otherwise specified.
- A.C. testing inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0".
- Timing measurements are made at 1.5V for both logic "1" and "0".
- Rise and Fall time of inputs and outputs signals are measured between 0.8V and 2.0V respectively unless otherwise specified.
- All timings are relative to CLK2 crossing the 1.5V level.
- All A.C. parameters are valid only after 100 μs from power up.

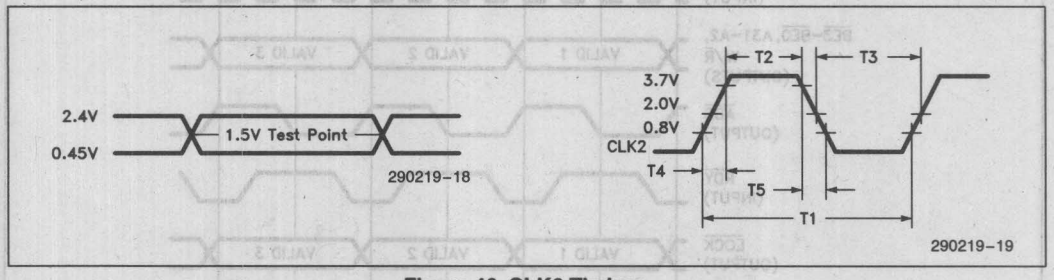
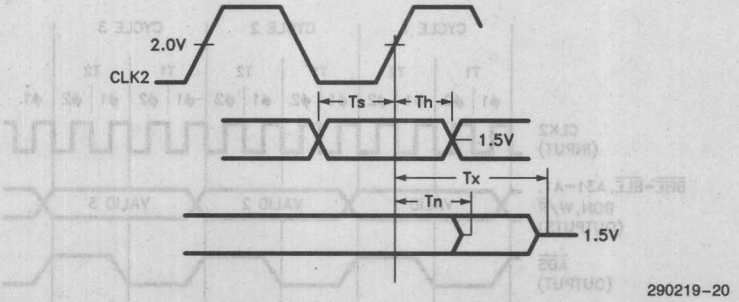


Figure 46. CLK2 Timings

Two types of timing specifications are presented below:

1. Input Timing—minimum setup and hold times.
2. Output Timings—output delays and float times from CLK2 rising edge.

Figure 45 defines how the measurements should be done:



LEGEND:
Ts = Input Setup Time
Th = Input Hold Time
Tn = Minimum output delay or Mininum float delay
Tx = Maximum output delay or Maximum float delay

Figure 47. Drive Levels and Measurements Points for A.C. Specifications

INPUT WAVEFORMS

Ts = T13, T15, T17, T19, T21, T23, T27, T29, T31
Th = T14, T16, T18, T20, T22, T22a, T24, T28, T30, T32

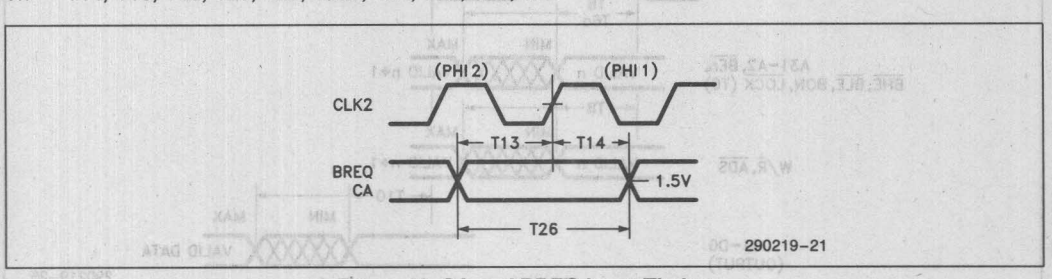


Figure 48. CA and BREQ Input Timing

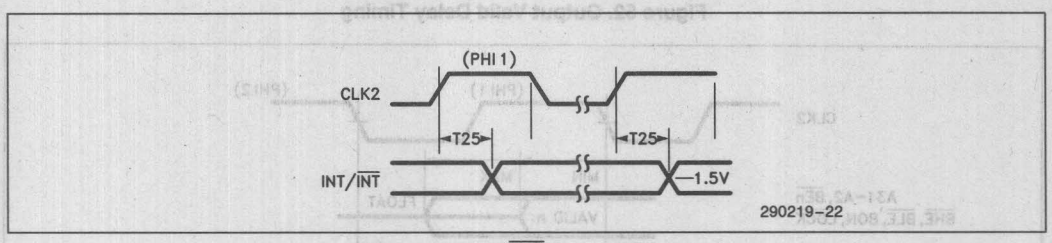


Figure 49. INT/INT Output Timing

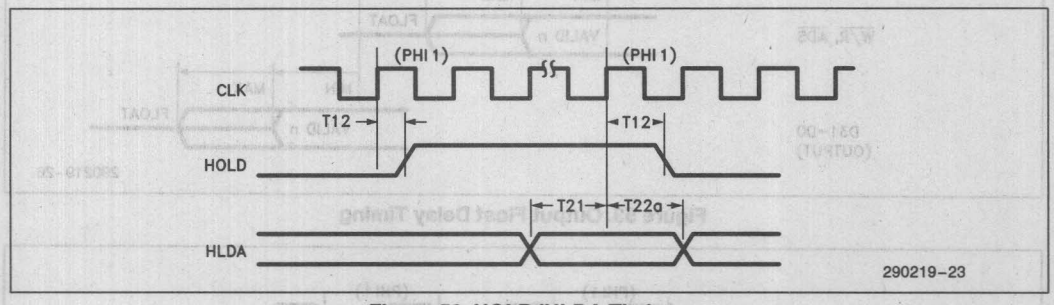


Figure 50. HOLD/HLDA Timings

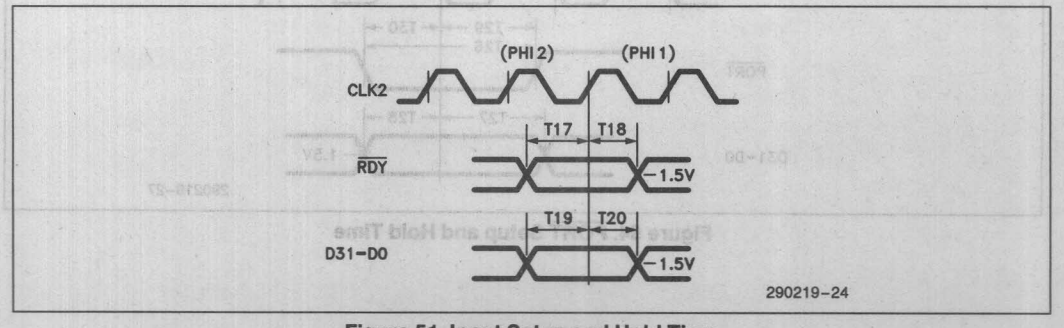


Figure 51. Input Setup and Hold Time

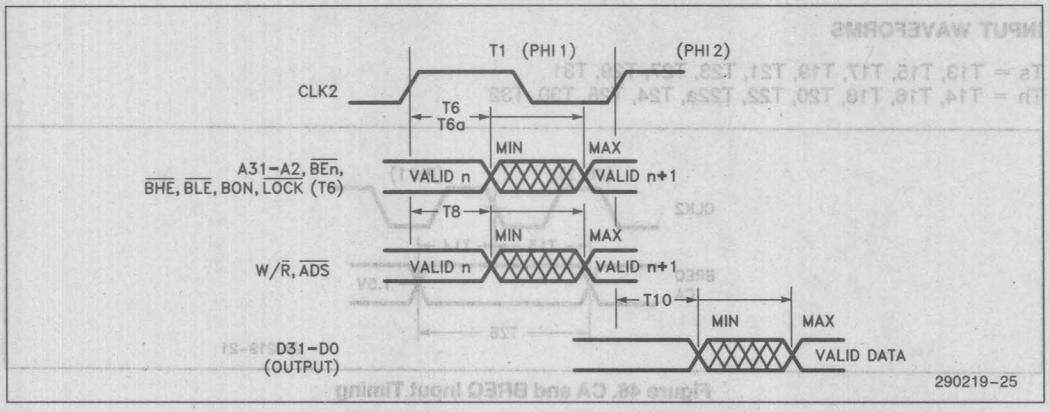


Figure 52. Output Valid Delay Timing

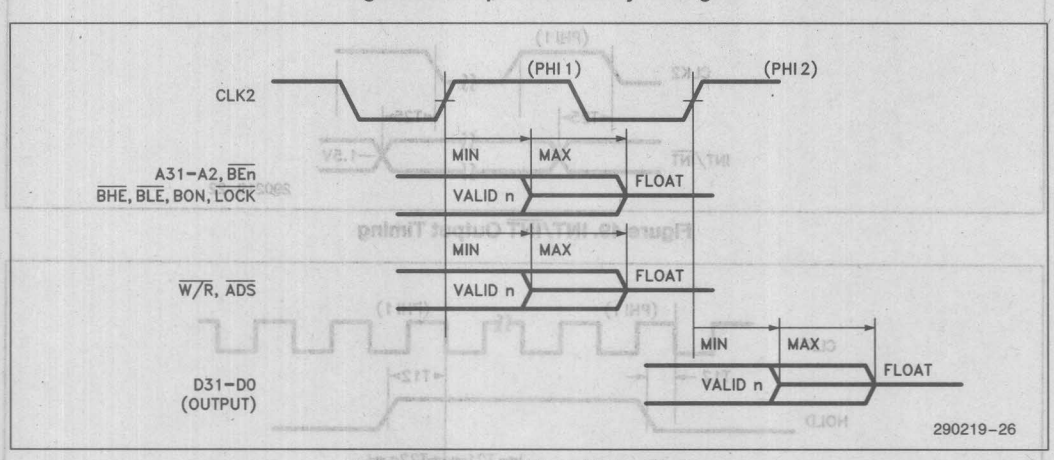


Figure 53. Output Float Delay Timing

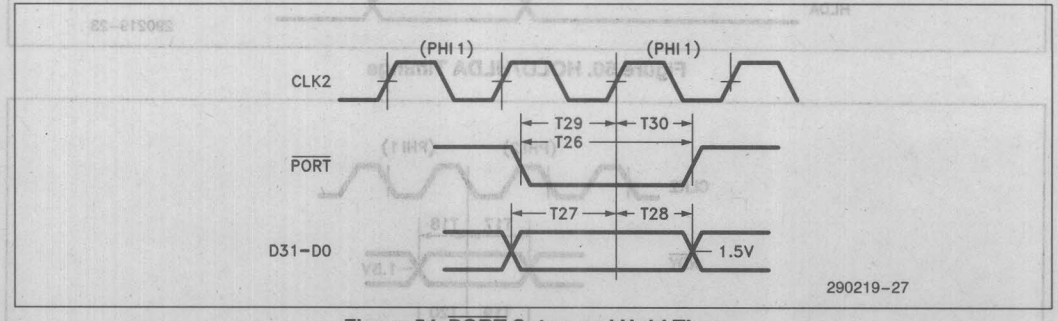


Figure 54. PORT Setup and Hold Time

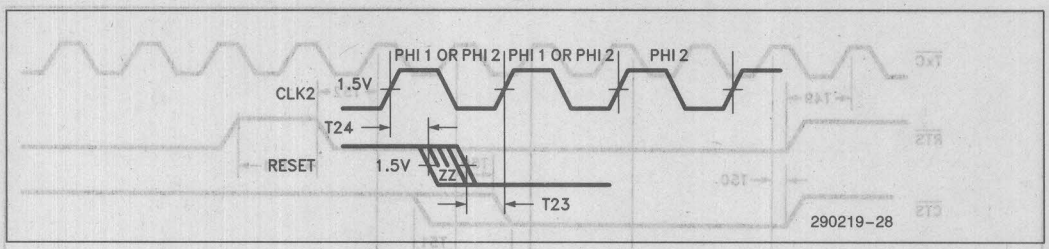


Figure 55. RESET Input Timing

SERIAL A.C. TIMING CHARACTERISTICS

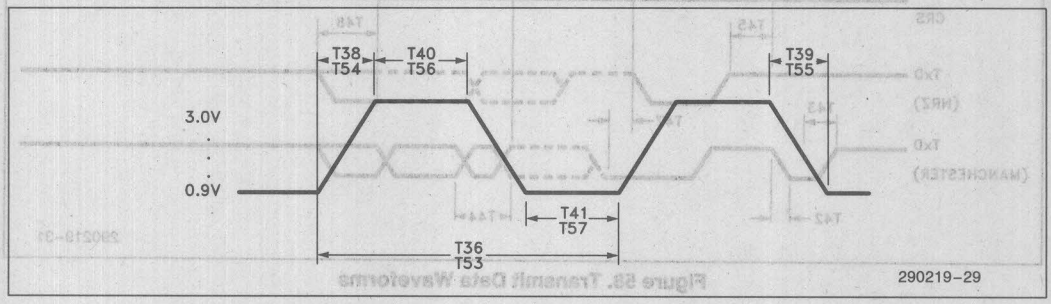


Figure 56. Serial Input Clock Timing

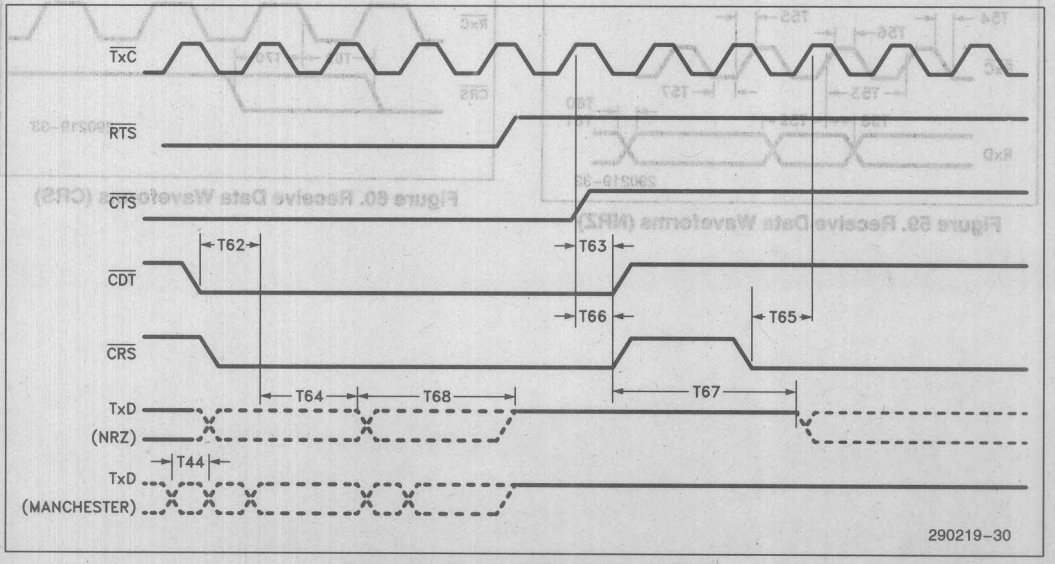


Figure 57. Transmit Data Waveforms

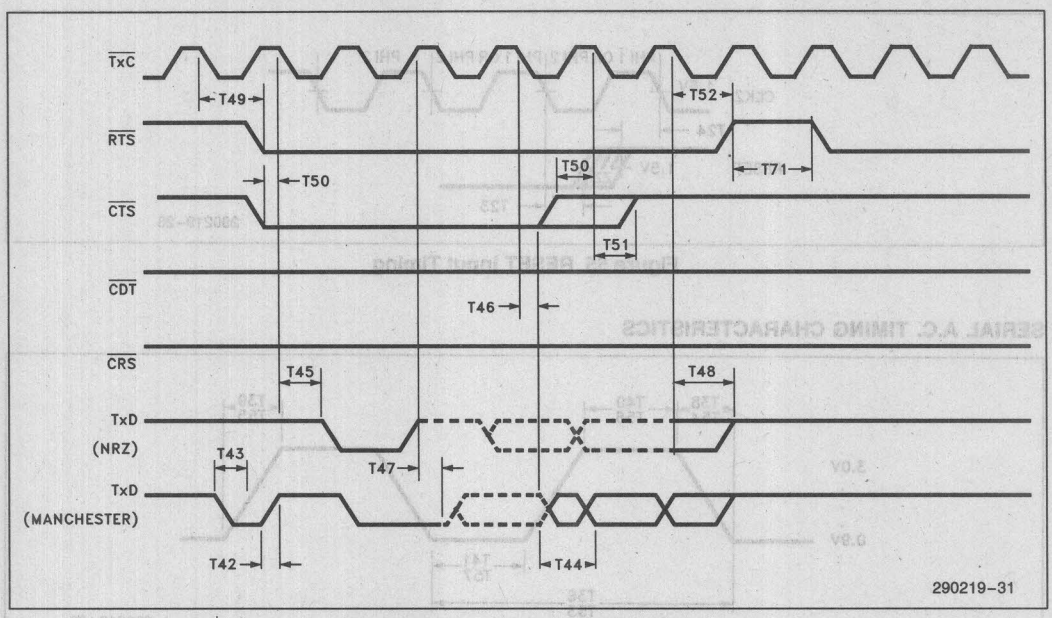


Figure 58. Transmit Data Waveforms

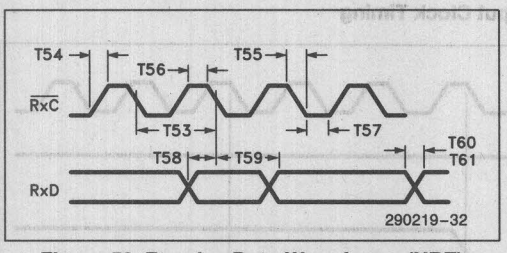


Figure 59. Receive Data Waveforms (NRZ)

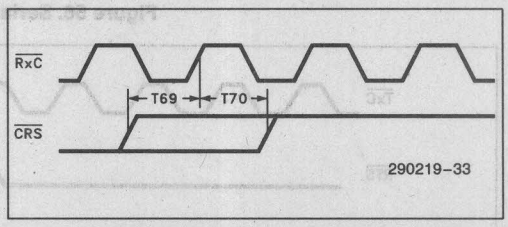
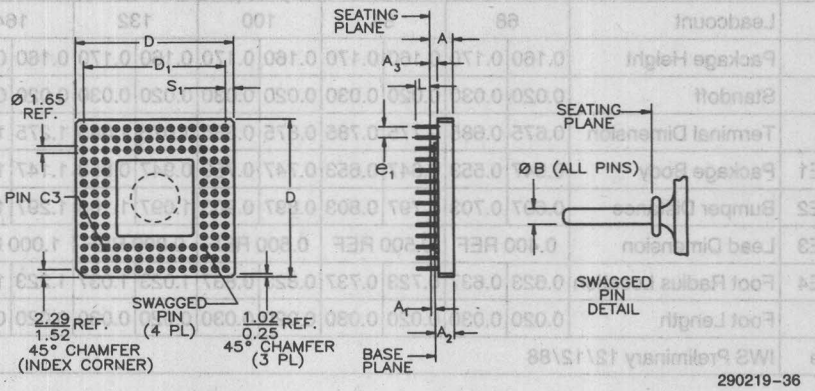


Figure 60. Receive Data Waveforms (CRS)

OUTLINE DIAGRAMS

132 LEAD CERAMIC PIN GRID ARRAY PACKAGE INTEL TYPE A



290219-36

Family: Ceramic Pin Grid Array Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
A	3.56	4.57		0.140	0.180	
A_1	0.76	1.27	Solid Lid	0.030	0.050	Solid Lid
A_2	2.67	3.43	Solid Lid	0.105	0.135	Solid Lid
A_3	1.14	1.40		0.045	0.055	
B	0.43	0.51		0.017	0.020	
D	36.45	37.21		1.435	1.465	
D_1	32.89	33.15		1.295	1.305	
e_1	2.29	2.79		0.090	0.110	
L	2.54	3.30		0.100	0.130	
N	132			132		
S_1	1.27	2.54		0.050	0.100	
ISSUE	IWS 10/12/88					

Intel Case Outline Drawings
Plastic Quad Flat Pack (PQFP)
0.025 Inch (0.635mm) Pitch

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170	0.160	0.170
A1	Standoff	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
D, E	Terminal Dimension	0.675	0.685	0.775	0.785	0.875	0.885	1.075	1.085	1.275	1.285	1.475	1.485
D1, E1	Package Body	0.547	0.553	0.647	0.653	0.747	0.753	0.947	0.953	1.147	1.153	1.347	1.353
D2, E2	Bumper Distance	0.697	0.703	0.797	0.803	0.897	0.903	1.097	1.103	1.297	1.303	1.497	1.503
D3, E3	Lead Dimension	0.400 REF		0.500 REF		0.600 REF		0.800 REF		1.000 REF		1.200 REF	
D4, E4	Foot Radius Location	0.623	0.637	0.723	0.737	0.823	0.837	1.023	1.037	1.223	1.237	1.423	1.437
L1	Foot Length	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030	0.020	0.030
Issue	IWS Preliminary 12/12/88												INCH

Symbol	Description	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
N	Leadcount	68		84		100		132		164		196	
A	Package Height	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32	4.06	4.32
A1	Standoff	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
D, E	Terminal Dimension	17.15	17.40	19.69	19.94	22.23	22.48	27.31	27.56	32.39	32.64	37.47	37.72
D1, E1	Package Body	13.89	14.05	16.43	16.59	18.97	19.13	24.05	24.21	29.13	29.29	34.21	34.37
D2, E2	Bumper Distance	17.70	17.85	20.24	20.39	22.78	22.93	27.86	28.01	32.94	33.09	38.02	38.18
D3, E3	Lead Dimension	10.16 REF		12.70 REF		15.24 REF		20.32 REF		25.40 REF		30.48 REF	
D4, E4	Foot Radius Location	15.82	16.17	18.36	18.71	21.25	21.25	25.89	26.33	31.06	31.41	36.14	36.49
L1	Foot Length	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76	0.51	0.76
Issue	IWS Preliminary 12/12/88												mm

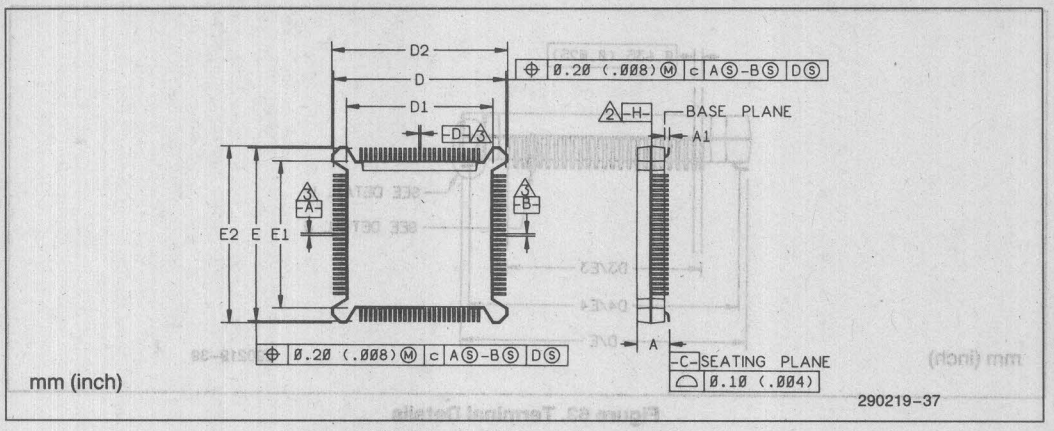


Figure 61. Principal Dimensions and Datums

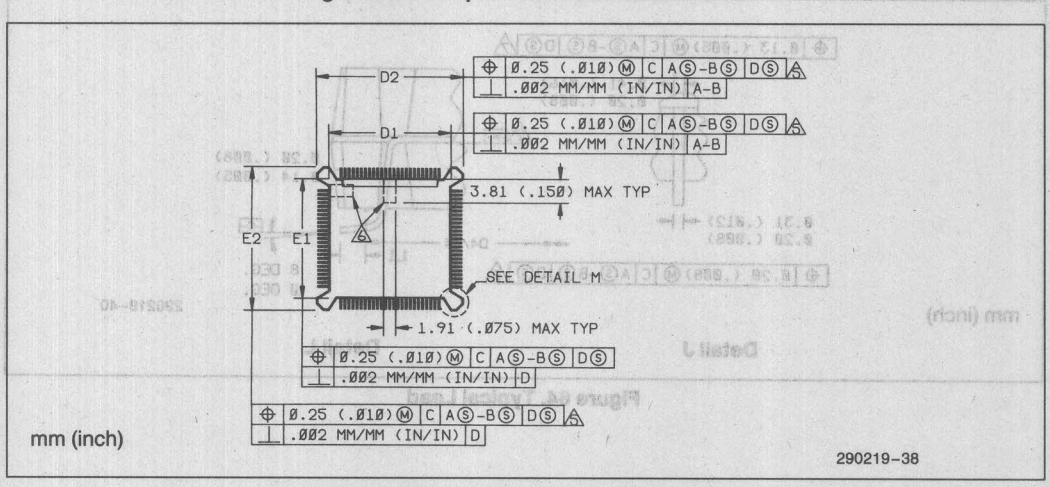


Figure 62. Molded Details

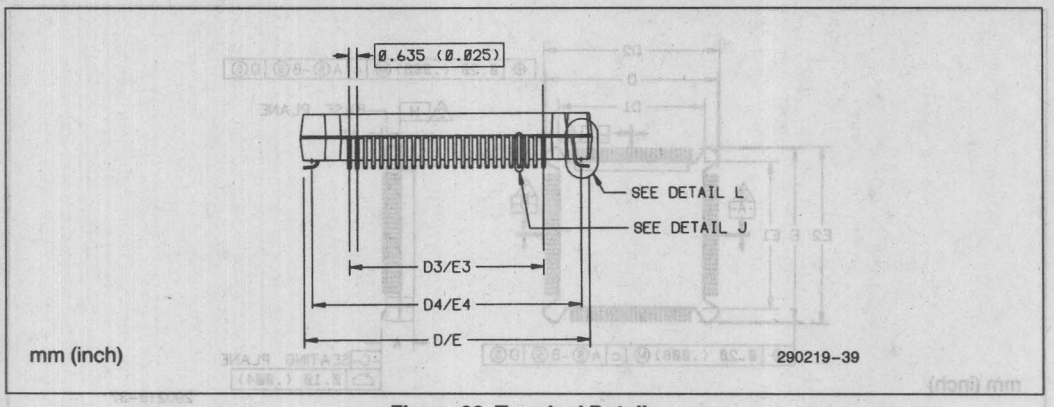


Figure 63. Terminal Details

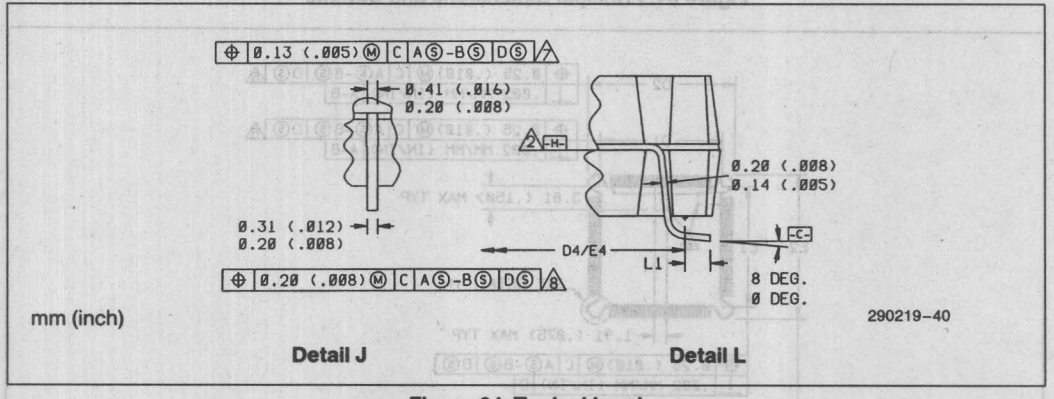


Figure 64. Typical Lead

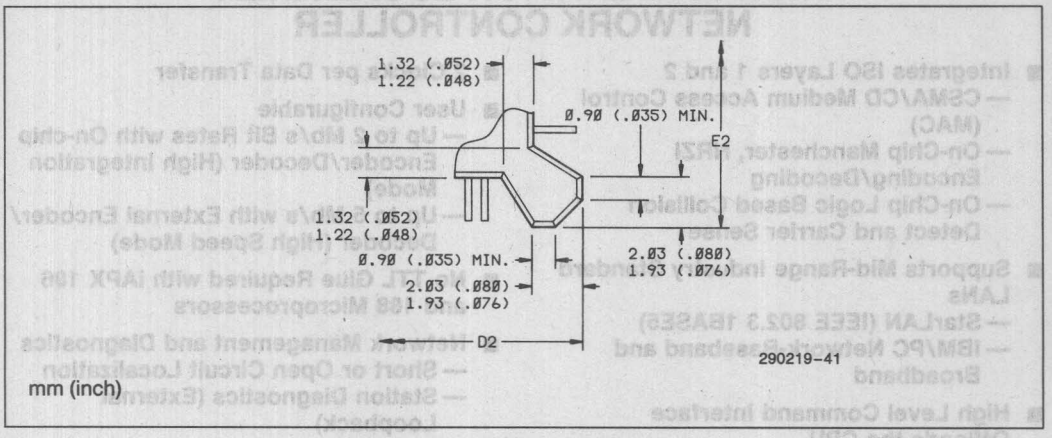


Figure 65. Detail M

REVISION HISTORY

The 82596 LAN Coprocessor data sheet version -003 contains updates and improvements to previous versions.

1. Updated A.C. Timings
2. Added Pin Cross Reference tables.
3. Added Bus Cycle figures.

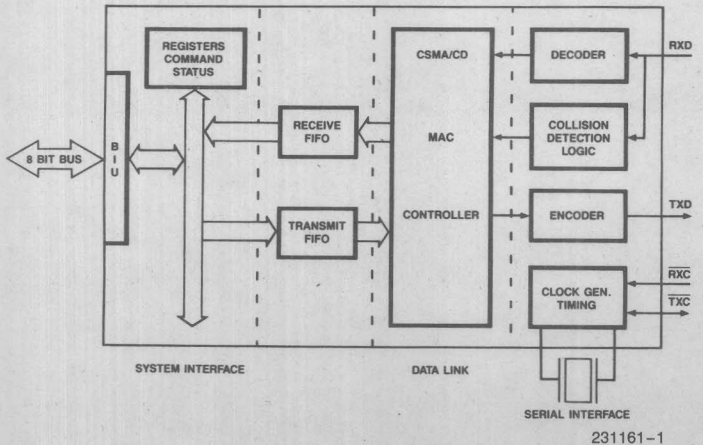
82588
HIGH INTEGRATION LOCAL AREA NETWORK CONTROLLER

- **Integrates ISO Layers 1 and 2**
  - **— CSMA/CD Medium Access Control (MAC)**
  - **— On-Chip Manchester, NRZI Encoding/Decoding**
  - **— On-Chip Logic Based Collision Detect and Carrier Sense**
- **Supports Mid-Range Industry Standard LANs**
  - **— StarLAN (IEEE 802.3 1BASE5)**
  - **— IBM/PC Network-Baseband and Broadband**
- **High Level Command Interface Offloads the CPU**
- **Efficient Memory Use Via Multiple Buffer Reception**
- **2 Clocks per Data Transfer**
- **User Configurable**
  - **— Up to 2 Mb/s Bit Rates with On-chip Encoder/Decoder (High Integration Mode)**
  - **— Up to 5 Mb/s with External Encoder/Decoder (High Speed Mode)**
- **No TTL Glue Required with iAPX 186 and 188 Microprocessors**
- **Network Management and Diagnostics**
  - **— Short or Open Circuit Localization**
  - **— Station Diagnostics (External Loopback)**
  - **— Self Test Diagnostics Internal Loopback User Readable Registers**

The 82588 is a highly integrated CSMA/CD controller designed for cost sensitive, mid-range Local Area Network (LAN) applications, such as personal computer networks.

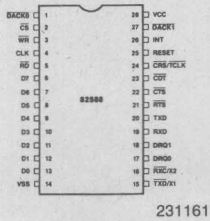
At data rates of up to 2 Mb/s, the 82588 provides a highly integrated interface and performs: CSMA/CD Data Link Control, Manchester, Differential Manchester or NRZI encoding/decoding, clock recovery; Carrier Sense, and Collision Detection. This mode is called "High Integration Mode." In the 82588 "High Speed Mode", the user can transfer data at a rate of up to 5 Mb/s. In this mode the physical link functions are done external to the 82588.

The 82588 is available in a 28 pin DIP and 44 lead PLCC package and fabricated in Intel's reliable HMOS II 5 volt technology.



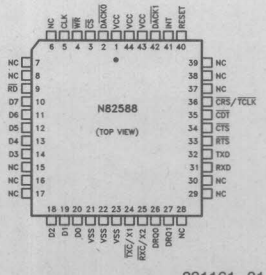
231161-1

Figure 1. 82588 Block Diagram



231161-2

Figure 2. 82588 Pin Configuration (DIP)



231161-31

Figure 3. 82588 Pin Configuration (PLCC)

September 1990
Order Number: 231161-005

Table 1. Pin Description

Symbol	Pin No. DIP	Pin No. PLCC	Type	Name and Function
D7 D6 D5 D4 D3 D2 D1 D0	6 7 8 9 10 11 12 13	10 11 12 13 14 18 19 20	I/O	**DATA BUS:** The Data Bus lines are bi-directional three state lines connected to the system's Data Bus for the transfer of data, commands, status and parameters.
$\overline{RD}$	5	9	I	**READ:** Together with $\overline{CS}$, $\overline{DACK0}$ or $\overline{DACK1}$, Read controls data or status transfers out of the 82588 registers.
$\overline{WR}$	3	4	I	**WRITE:** Together with $\overline{CS}$, $\overline{DACK0}$ or $\overline{DACK1}$, Write controls data or command transfers into the 82588 registers.
$\overline{CS}$	2	3	I	**CHIP SELECT:** When this signal is LOW, the 82588 is selected by the CPU for transfer of command or status. The direction of data flow is determined by the $\overline{RD}$ or $\overline{WR}$ inputs.
CLK	4	5	I	**CLOCK:** System clock. TTL compatible signal.
RESET	25	40	I	**RESET:** A HIGH signal on this pin will cause the 82588 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock cycles.
INT	26	41	O	**INTERRUPT:** Active HIGH signal indicates to the CPU that the 82588 is requesting an interrupt.
DRQ0	17	26	O	**DMA REQUEST (CHANNEL 0):** This pin is used by the 82588 to request a DMA transfer. DRQ0 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active for multiple transfers.
DRQ1	18	27	O	**DMA REQUEST (CHANNEL 1):** This pin is used by the 82588 to request a DMA transfer. DRQ1 remains HIGH as long as 82588 requires data transfers. Burst transfers are done by having the signal active or multiple transfers.
$\overline{DACK0}$	1	2	I	**DMA ACKNOWLEDGE (CHANNEL 0):** When LOW, this input signal from the DMA Controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 0.
$\overline{DACK1}$	27	42	I	**DMA ACKNOWLEDGE (CHANNEL 1):** When LOW, this input signal from the DMA controller notifies the 82588 that the requested DMA cycle is in progress. This signal acts like chip select for data and parameter transfer using DMA channel 1.

Table 1. Pin Description (Continued)

Symbol	Pin No. DIP	Pin No. PLCC	Type	Name and Function
				High Integration Mode
X1/X2	15/16	24/25	I	**OSCILLATOR INPUTS:** These inputs may be used to connect a quartz crystal that controls the internal clock generator for the serial unit. X1 may also be driven by a MOS level clock whose frequency is 8 or 16 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 has an external MOS clock.
				High Speed Mode
$\overline{TxC}$	15	24	I	**TRANSMIT CLOCK:** This signal provides timing information to the internal serial logic, depending upon the mode of data transfer. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding the transmitted bit center is aligned with the $\overline{TxC}$ LOW to HIGH transition.
$\overline{RxC}$	16	25	I	**RECEIVE CLOCK:** This signal provides timing information to the internal serial logic. NRZ data should be provided for reception (RxD). The state of the RxD pin is sampled on the HIGH to LOW transition of $\overline{RxC}$. The operating mode of the 82588 is defined when configuring the chip.
$\overline{TCLK}/\overline{CRS}$	24	36	I	In High Speed Mode, this pin is Carrier Sense, input CRS, and is used to notify the 82588 that there is activity on the serial link.
			O	In High Integration Mode, this pin is Transmit Clock, $\overline{TCLK}$, and is used to output the transmit clock.
$\overline{CDT}$	23	35	I	**COLLISION DETECT:** This input notifies the 82588 that a collision has occurred. It is sensed only if the 82588 is configured for external Collision Detect (external circuitry is then required for detecting the collision).
RxD	19	31	I	**RECEIVE DATA:** This pin receives serial data.
TxD	20	32	O	**TRANSMIT DATA:** This pin transmits data to the Serial Link. This signal is HIGH when not transmitting.
$\overline{RTS}$	21	33	O	**REQUEST TO SEND:** When this signal is LOW, the 82588 notifies an external interface that it has data to transmit. It is forced HIGH after a reset and when transmission is stopped.
$\overline{CTS}$	22	34	I	**CLEAR TO SEND:** CTS enables the 82588 to start transmitting data. Raising this signal to HIGH stops the transmission.
VCC	28	1, 43, 44		**POWER:** +5V Supply
VSS	14	21, 22, 23		Ground

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
NC		6		**NO CONNECT:** These pins are reserved for future use.
		7		
		8		
		15		
		16		
		17		
		28		
		29		
		30		
		37		
		38		
		39		

FUNCTIONAL DESCRIPTION

High Integration Mode

The 82588 LAN Controller is a highly integrated CSMA/CD controller for cost sensitive LAN applications such as personal computer networks. Included on chip is a programmable CSMA/CD controller, an NRZI and Manchester encoder/decoder with clock recovery, and two collision detection mechanisms. With the addition of simple transceiver line drivers or RF Modem, the 82588 performs all the major functions of the ISO Physical and Data Link Layers.

CSMA/CD Controller

The 82588 on-chip CSMA/CD controller is programmable, which allows it to operate in a variety of LAN environments, including industry standards such as StarLAN (IEEE 802.3 1BASE5) and the 2 Mb/s IBM PC Network (both baseband and broadband transmission). Programmable parameters include:

- Framing (End of Carrier of SDLC)
- Address field length
- Station priority
- Interframe spacing
- Slot time
- CRC-32 OR CRC-16

Encoder/Decoder

The on-chip NRZI and Manchester encoder/decoder supports data rates up to 2 Mb/s. Manchester encoding is typically used in baseband applications and NRZI is used in broadband applications.

Collision Detection

One of the 82588's unique features is its on-chip logic based collision detection. To ensure a high probability of collision detection two mechanisms are provided. The Code Violation method defines a collision when a transition edge occurs outside the area of normal transitions as specified by either the Manchester or NRZI encoding methods. Bit Comparison method compares the signature of the transmitted frame to the received frame signature (re-calculated by the 82588 while listening to itself). If the signatures are identical the frame is assumed to have been transmitted without a collision.

System Interface

In addition to providing the functions necessary for interfacing to the LAN, the 82588 has a friendly system interface that eases the design effort. First, the 82588 has a high level command interface; that is the CPU sends the 82588 commands such as Transmit or Configure. This means the designer does not have to write low level software to perform these tasks, and it offloads the CPU in the application. Second, the 82588 supports an efficient memory structure called Multiple Buffer Reception in which buffers are chained together while receiving frames. This is an important feature in applications with limited memory, such as personal computers. Third, the 82588 has two independent sixteen byte FIFO's, one for reception and one for transmission. The FIFO's allow the 82588 to tolerate bus latency. Finally the 82588 provides an eight byte data path that supports up to 4 Mbytes/second using external DMA.

Network Management & Diagnostics

The 82588 provides a rich set of diagnostic and network management functions including: internal and external loopback, channel activity indicators, optional capture of all frames regardless of destination address (Promiscuous Mode), capture of collided frames, (if address matches), and time domain reflectometry for locating fault points in the network cable. The 82588 register Dump command ensures reliable software by dumping the content of the 82588 registers into the system memory.

The next section will describe the 82588 system bus interface, the 82588 network interface, and the 82588 internal architecture.

82588/Host CPU Interaction

The CPU communicates with the 82588 through the system's memory and 82588's on-chip registers. The CPU creates a data structure in the memory, programs the external DMA controller with the start address and byte count of the block, and issues the command to the 82588.

The 82588 is optimized for operating with the iAPX 186/188, but due to the small number of hardware signals between the 82588 and the CPU, the 82588 can operate easily with other processors. The data bus is 8 bits wide and there is no address bus.

Chip Select and Interrupt lines are used to communicate between the 82588 and the host as shown in the Figure 3. Interrupt is used by the 82588 to draw the CPU's attention. The Chip Select is used by the CPU to draw the 82588's attention.

There are two kinds of transfer over the bus: Command/Status and data transfers. Command/Status transfers are always performed by the CPU. Data transfers are requested by the 82588, and are typically performed by a DMA controller. The table given in Figure 4 shows the Command/Status and data transfer control signals.

The CPU writes to 82588 using $\overline{CS}$ and $\overline{WR}$ signals. The CPU reads the 82588 status register using $\overline{CS}$ and $\overline{RD}$ signals.

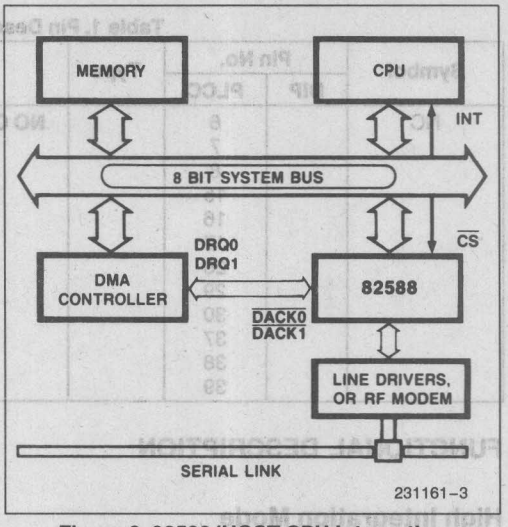


Figure 3. 82588/HOST CPU Interaction

To initiate an operation like Transmit or Configure (see Figure 5), a Write command from CPU to 82588 is issued by the CPU. A Read operation from CPU gives the status of the 82588. Although there are four status registers they're read using the same port in a round robin fashion (Figure 6).

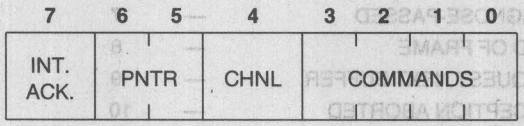
Any parameters or data associated with a command are transferred between the memory and 82588 using DMA. The 82588 has two data channels, each having Request and Acknowledge lines. Typically one channel is used to receive data and other to transmit data and perform all the other initialization and maintenance operations like Configure, Address Set-Up, Diagnose, etc. The channels are identical and can be used interchangeably.

When the 82588 requires access to the memory for parameter or data transfer it activates the DMA request lines and uses the DMA controller to achieve the data transfer. Upon the completion of an operation, the 82588 interrupts the CPU. The CPU then reads results of the operation (the status of the 82588).

Pin Name			Function
$\overline{CS}$*	$\overline{RD}$	$\overline{WR}$	
1	x	x	No transfer to/from Command/Status
0	1	1	
0	0	0	Illegal
0	0	1	Read from status register
0	1	0	Write to Command register
$\overline{DACK0}$[$\overline{DACK1}$]*	$\overline{RD}$	$\overline{WR}$	
1	x	x	No DMA transfer
0	1	1	
0	0	0	Illegal
0	0	1	Data Read from DMA channel 0 [or 1]
0	1	0	Data Write to DMA channel 0 [or 1]

* Only one of $\overline{CS}$, $\overline{DACK0}$ and $\overline{DACK1}$ may be active at any time.

Figure 4. Databus Control Signals and Their Functions



COMMAND REGISTER

COMMANDS		VALUE	COMMANDS		VALUE
NOP	—	0	ABORT	—	13
IA-SETUP	—	1	RECEIVER-ENABLE	—	8
CONFIGURE	—	2	ASSIGN NEXT BUF	—	9
MC-SETUP	—	3	RECEIVE-DISABLE	—	10
TRANSMIT	—	4	STOP-RECEPTION	—	11
TDR	—	5	RESET	—	14
DUMP	—	6	FIX PTR	—	15 (CHNL = 1)
DIAGNOSE	—	7	RLS PTR	—	15 (CHNL = 0)
RETRANSMIT	—	12			

Figure 5. Command Format and Operation Values

	7	6	5	4	3	2	1	0
Status 0	INT	RCV	EXEC	CHNL	EVENT			
Status 1	RESULT 1							
Status 2	RESULT 2							
Status 3	RCV CHNL	RCV STATE		BUFF CHNG NO. OF BUF		EXEC CHNL	EXEC STATE	

EVENTS	VALUE (STATUS 0)
IA-SETUP-DONE	1
CONFIGURE-DONE	2
MC-SETUP-DONE	3
TRANSMIT-DONE	4
TDR-DONE	5
DUMP-DONE	6
DIAGNOSE-PASSED	7
END OF FRAME	8
REQUEST NEXT BUFFER	9
RECEPTION ABORTED	10
RETRANSMIT-DONE	12
EXECUTION-ABORTED	13
DIAGNOSE-FAILED	15

Figure 6. Status Registers and Event Values

Transmitting a Frame

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 7. Its first two bytes specify the length of the rest of the block. The next few bytes (Up to 6 bytes long) contain the destination address of the node it is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block and other control information and then issues the Transmit command to the 82588.

Upon receiving the command, the 82588 fetches the first two bytes of the block to determine the length of the block. If the link is free, and the first data byte was fetched, the 82588 begins transmitting the preamble and concurrently fetches the bytes from the Transmit Data Block and loads them into a 16 byte FIFO to keep them ready for transmitting. The FIFO is a buffer between the serial and parallel part of the 82588. The on-chip FIFOs help the 82588 to tolerate system bus latency as well as provide efficient usage of system bandwidth.

The destination address is sent out after the preamble. This is followed by the source or the station individual address, which is stored earlier on the 82588 using the IA-SETUP command. After that, the entire information field is transmitted followed by a CRC field calculated by the 82588. If during the transmission of the frame, a collision is encountered, then the transmission is aborted and a jam pattern is sent out after completion of the preamble. The 82588 generates an Interrupt indicating the experience of a collision and the frame has to be re-transmitted. Retransmission is done by the CPU exactly as the Transmit command except the Re-Transmit command keeps track of the number of collisions encountered. When the 82588 gets the Retransmit command and the exponential back-off time is expired, the 82588 transmits the frame again. The transmitted frame can be coded to either Manchester, Differential Manchester or NRZI methods.

Collision Detection

The 82588 eliminates the need for external collision detection logic, in most applications, while easing or eliminating the need for complex transceivers. Two algorithms are used for collision detection: Bit Comparison and Code Violation. The Bit Comparison Method is useful in Broadband networks where there are separate transmit and receive channels. Bit Comparison compares the "signature" of the transmitted data and received data at the end of the collision window in any network configuration. This algorithm calculates the CRC after a programmable number of transmitted bits, holds this CRC in a register, and compares it with received data's CRC. A CRC or "signature" difference indicates a collision. The code violation is detected if the encoding of the received data has any bit that does not fit the encoding rules. The code violation method is useful in short bus topology and serial backplane applications where bit attenuation over the bus is negligible.

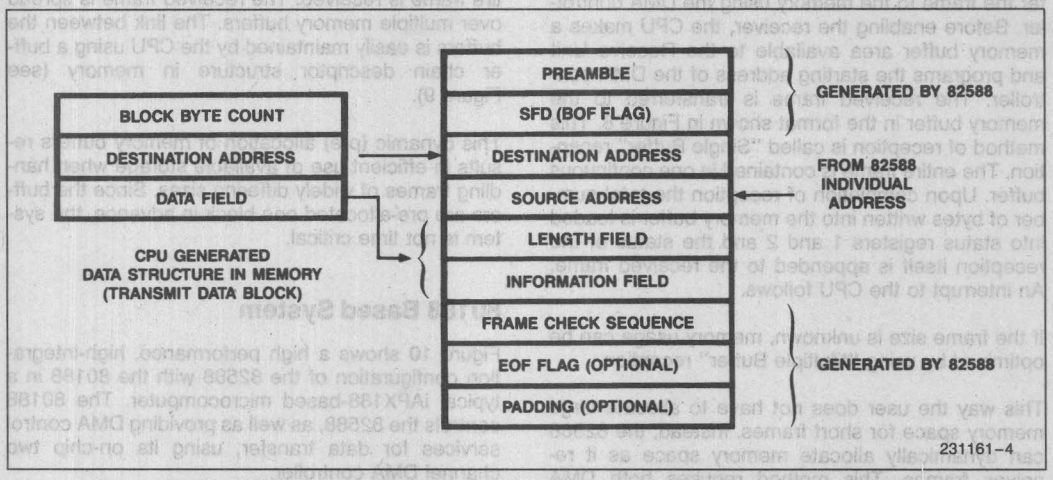


Figure 7. The 82588 Frame Structure and location of Data element in System Memory

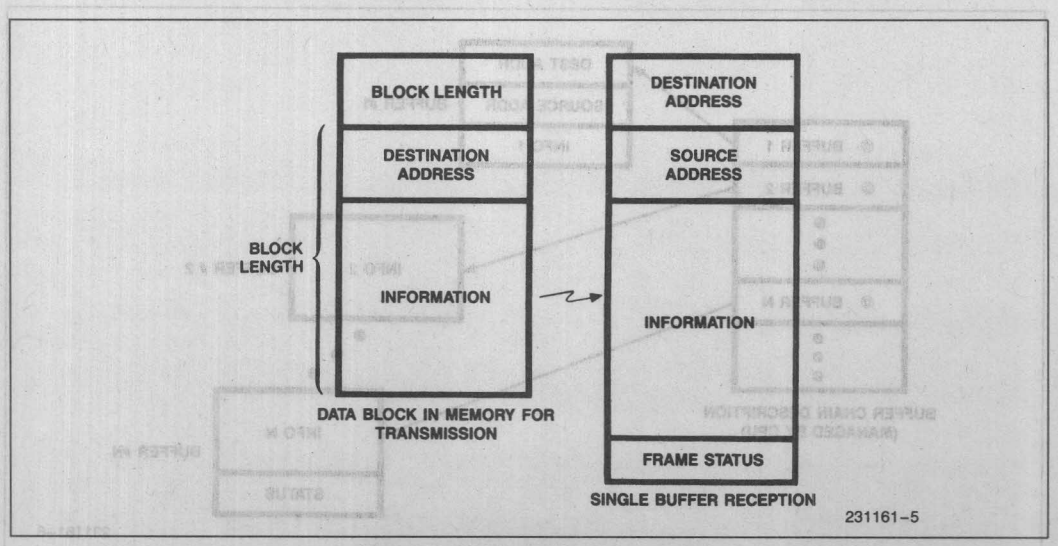


Figure 8. Single Buffer Reception

Receiving a Frame

The 82588 can receive a frame when its receiver has been enabled. The received frame is decoded by either on-chip Manchester, Differential Manchester or NRZI decoders in High Integration Mode and NRZI in High Speed Mode. The 82588 checks for an address match for an individual address, a Mulitcast address or a Broadcast address. In the Promiscuous mode the 82588 receives all frames. Only when the address match is successful does the 82588 transfer the frame to the memory using the DMA controller. Before enabling the receiver, the CPU makes a memory buffer area available to the Receive Unit and programs the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 8. This method of reception is called "Single Buffer" reception. The entire frame is contained in one continuous buffer. Upon completion of reception the total number of bytes written into the memory buffer is loaded into status registers 1 and 2 and the status of the reception itself is appended to the received frame. An interrupt to the CPU follows.

If the frame size is unknown, memory usage can be optimized by using "Multiple Buffer" reception.

This way the user does not have to allocate large memory space for short frames. Instead, the 82588 can dynamically allocate memory space as it receives frames. This method requires both DMA channels alternately to receive the frame. As the frame reception starts, the 82588 interrupts the CPU and automatically requests assignment of the next sequential buffer. The CPU does this and loads the second DMA channel with the next buffer information so that the 82588 can immediately switch to the other channel as soon as the current buffer is full. When the 82588 switches from the first to the second buffer it again interrupts the CPU requesting it to allocate another buffer on the other (previous) channel in advance. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU using a buffer chain descriptor structure in memory (see Figure 9).

This dynamic (pre) allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes. Since the buffers are pre-allocated one block in advance, the system is not time critical.

80188 Based System

Figure 10 shows a high performance, high-integration configuration of the 82588 with the 80188 in a typical iAPX188-based microcomputer. The 80188 controls the 82588, as well as providing DMA control services for data transfer, using its on-chip two channel DMA controller.

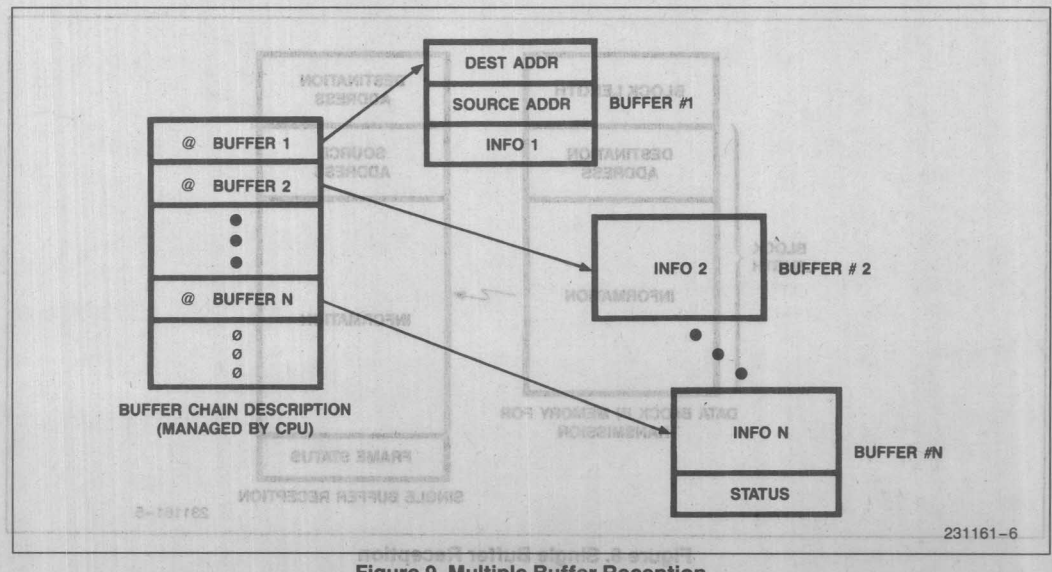


Figure 9. Multiple Buffer Reception

Link Interface

The Serial Interface Mode configuration parameter selects either a highly integrated Direct Link interface (High Integration Mode) or a highly flexible Transceiver Interface (High Speed Mode).

Application

In the High Integration Mode it is possible to connect the 82588 on a very short "Wired OR" link, on a longer twisted pair cable, or a broadband connection.

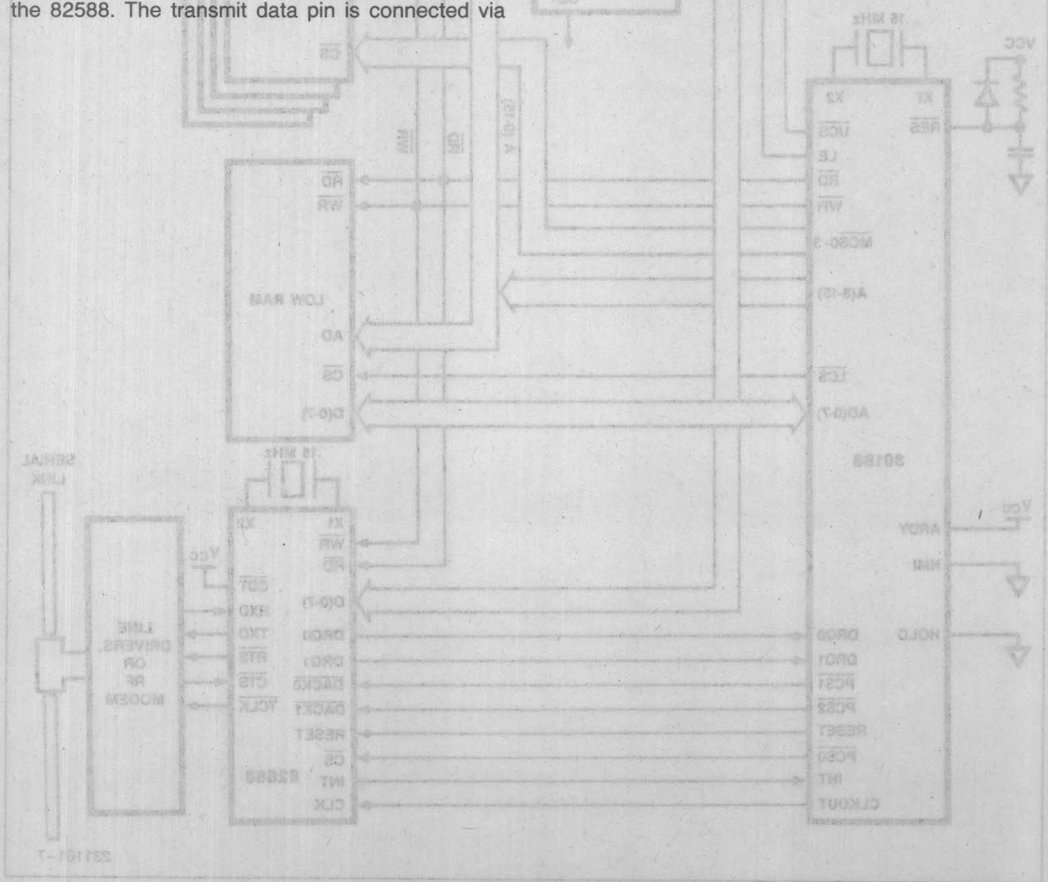
Twisted Pair Connection

The link consists of a twisted pair that interconnects the 82588. The transmit data pin is connected via a driver and the receive data pin is connected via a buffer. The twisted pair must be properly terminated to prevent reflections.

In the minimum configuration, TxD and RxD are connected to the twisted pair and $\overline{CTS}$ is grounded. The 82588 may control the driver with the $\overline{RTS}$ pin. It is also possible to use external circuitry for performing collision detection, and feeding it to the 82588 through the $\overline{CDT}$ pin.

Broadband Connection

The 82588 supports data communications over a broadband link in both its modes. Proper MODEM interface should be provided. Collision Detection by Bit Comparison, in High Interface Mode, can be applied to transmission over broadband links.



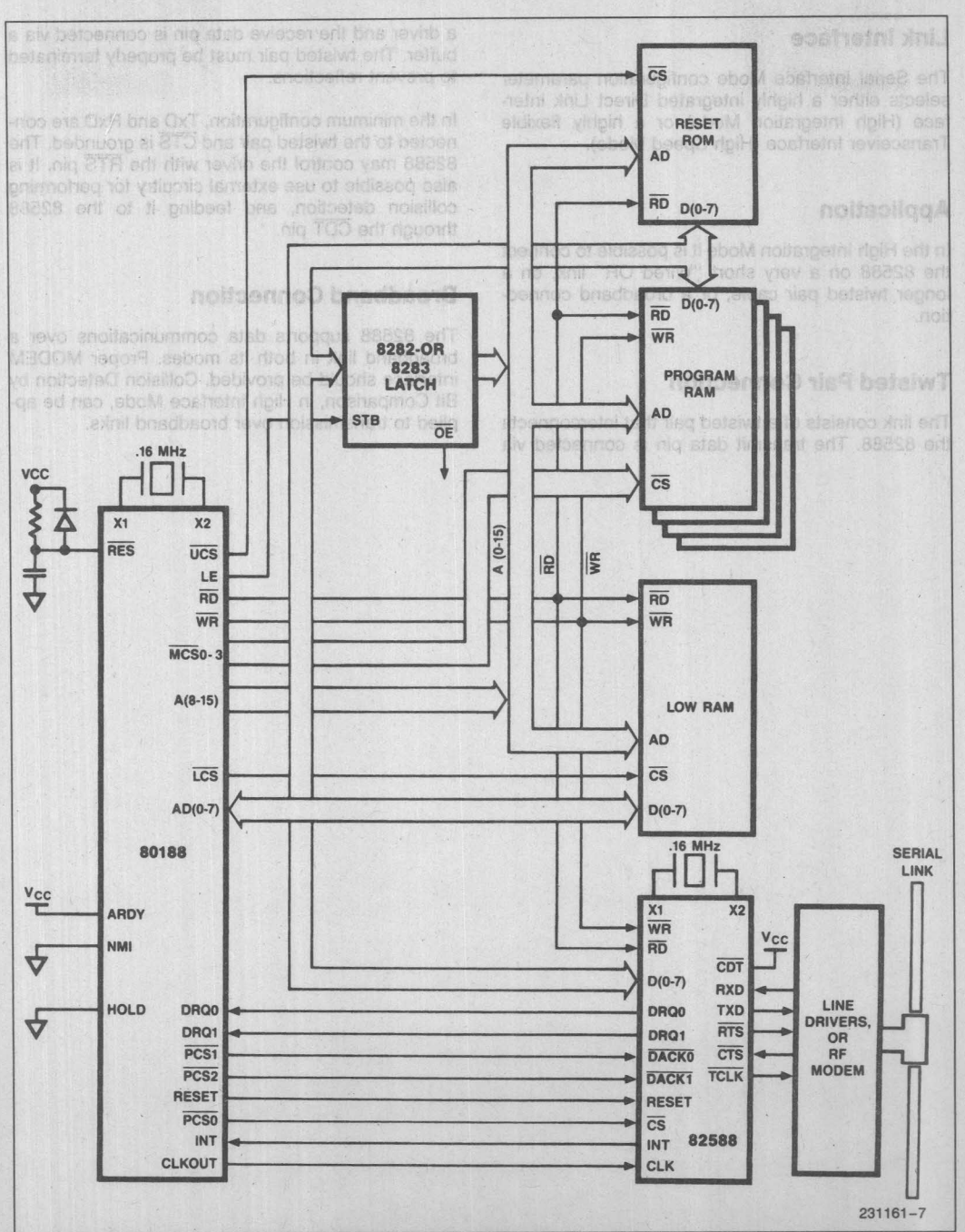


Figure 10. 80188 Based System

Absolute Maximum Ratings*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With
Respect to Ground . . . . . −1.0V to 7V
Power Dissipation . . . . . 1.7 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. Characteristics

(T_A = 0°C to +70°C; T_C (DIP) = 52°C to 108°C, T_C (PLCC) = 63°C to 116°C; VCC = +5V ±10%)

$\overline{TxC}$, $\overline{RxC}$ have MOS levels (See VMIL, VMIH). All other signals have TTL levels (See VIL, VIH, VOL, VOH).

Symbol	Parameter	Min	Max	Units	Test Conditions
VIL	Input Low Voltage (TTL)	−0.5	+0.8	V	
VIH	Input High Voltage (TTL)	2.0	VCC + 0.5	V	
VOL	Output Low Voltage (TTL)		0.45	V	IOL = 2.0 mA
VOH	Output High Voltage (TTL)	2.4		V	IOH = −400 µA
VMIL	Input Low Voltage (MOS)	−0.5	0.6	V	
VMIH	Input High Volatge (MOS)	3.9	VCC + 0.5	V	
ILI	Input Leakage Current		+10	µA	0 = VIN = VCC
ILO	Output Leakage Current		±10	µA	0.45 = VOUT = VCC
ICC	Power Supply Current		400 300	mA mA	T_A = 0°C T_A = +70°C

A.C. Characteristics

(T_A = 0°C to +70°C; T_C (DIP) = 52°C to 108°C, T_C (PLCC) = 63°C to 116°C; VCC = +5V ±10%)

System Clock Parameters

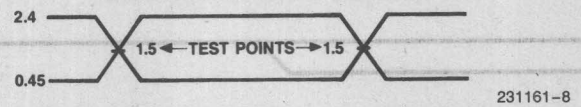
Symbol	Parameter	Min	Max	Units	Test Conditions
T1	CLK Cycle Period	125		ns	
T2	CLK Low Time	53	1000	ns	*5
T3	CLK High Time	53		ns	*6
T4	CLK Rise Time		15	ns	*1
T5	CLK Fall Time		15	ns	*2

A.C. Characteristics (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Reset Parameters					
T6	Reset Active to Clock Low	20		ns	*3
T8	Reset Pulse Width	4T1		ns	
T9	Control Inactve After Reset		T1	ns	
Interrupt Timing Parameters					
T10	CLK High to Interrupt Active		85	ns	*4
T11	$\overline{WR}$ Idle to Interrupt Idle		85	ns	*4
Write Parameters					
T12	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Setup to $\overline{WR}$ Low	0		ns	
T13	$\overline{WR}$ Pulse Width	95		ns	
T14	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Hold After $\overline{WR}$ High	0		ns	
T15	Data Setup to $\overline{WR}$ High	75		ns	
T16	Data Hold After $\overline{WR}$ High	0		ns	
Read Parameters					
T17	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Setup to $\overline{RD}$ Low	0		ns	
T18	$\overline{RD}$ Pulse Width	95		ns	
T19	$\overline{CS}$ or $\overline{DACK0}$ or $\overline{DACK1}$ Address Valid After $\overline{RD}$ High	0		ns	
T20	$\overline{RD}$ Low to Data Valid		80	ns	*7
T21	Data Float After $\overline{RD}$ High		55	ns	*7
DMA Parameters					
T22	CLK Low to DRQ0 or DRQ1 Active		85	ns	*4
T23	$\overline{WR}$ or $\overline{RD}$ Low to DRQ0 or DRQ1 Inactive		60	ns	*4

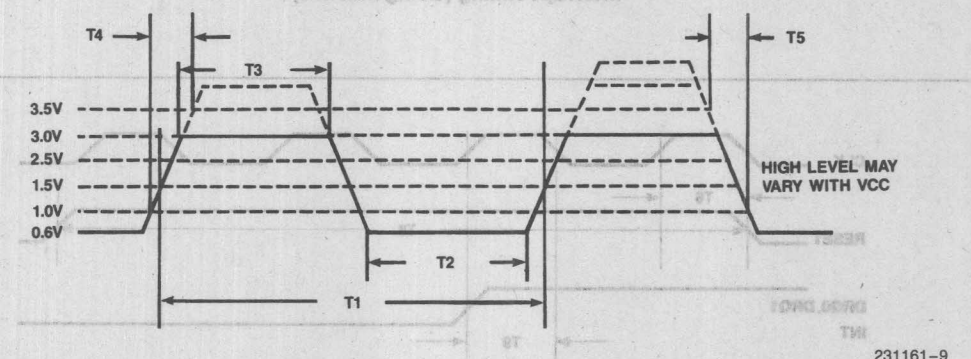
NOTES:
*1—0.8V–2.0V
*2—2.0V–0.8V
*3—to guarantee recognition at next clock
*4—CL = 50 pF
*5—measured at 1.5V
*6—measured at 1.5V
*7—CL = 20 pF–200 pF

A.C. TESTING INPUT/OUTPUT WAVEFORM



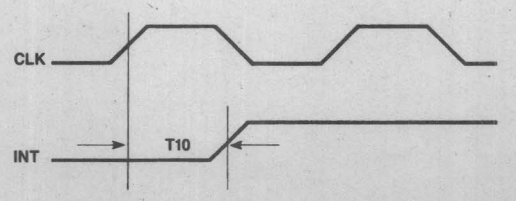
AC Testing Inputs are Driven at 2.4V for a Logic 1 and 0.45V for a Logic Q Timing Measurements are Made at 1.5V for Both a Logic 1 and Q:
Rise and Fall Time of Input/Output Signals are Measured Between 0.8V to 2.0V Respectively..

TTL Input/Output Voltage Levels for Timing Measurements

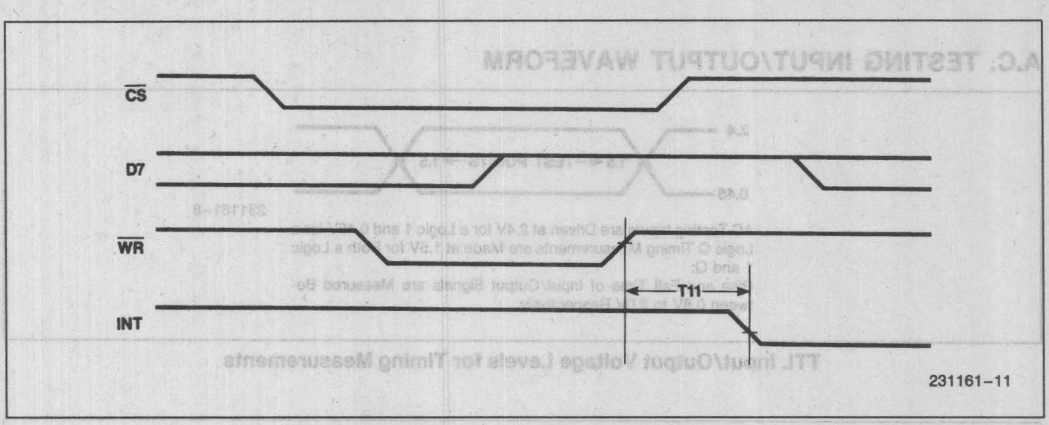


Rise and Fall Time of Input Signals are Measured Between 1.0V to 3.5V Respectively.

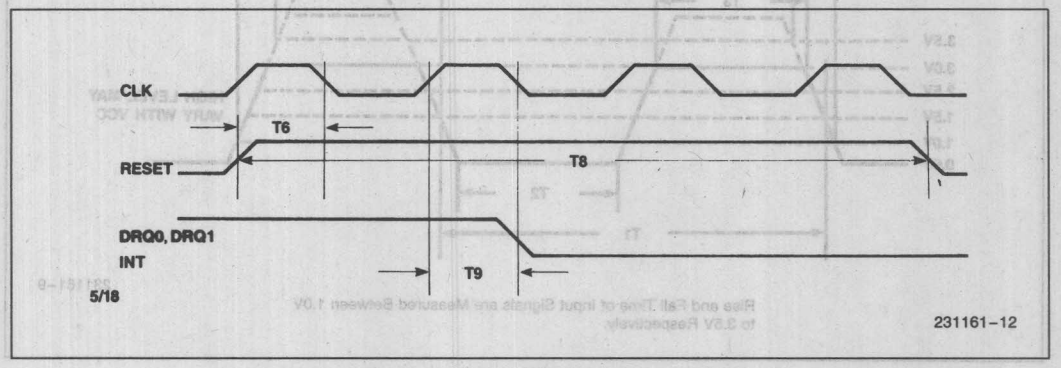
Clocks MOS Input Voltage Levels for Timing Measurements



Interrupt Timing (Going Active)



Interrupt Timing (Going Inactive)



Reset Timing

Serial Interface A.C. Timing Characteristics
High Integration Mode

$\overline{TFC}$ is the crystal or serial clock input at the X1 pin. When a serial clock is provided at the X1 pin, the maximum capacitive load allowed on the X2 pin is 15 pF.
$\overline{TFC}$ Frequency Range:

For Oscillator Frequency = 1 to 16 MHz (High)		
	×8 Sampling	×16 Sampling
$\overline{TCLK}$ Frequency	0.125 − 2 MHz	62.5 kHz − 1 MHz
T29 = $\overline{TCLK}$ Cycle Time	8 × T24	16 × T24
T30 = $\overline{TCLK}$ High Time	T24 (Typically)	T24 (Typically)
T31 = $\overline{TCLK}$ Low time	7 × T24 (Typically)	15 × T24 (Typically)

For Oscillator Frequency = 0 to 1 MHz (Low)*		
	×8 Sampling	×16 Sampling
$\overline{TCLK}$ Frequency	0 − 0.125 MHz	0 − 6.25 kHz
T29 = $\overline{TCLK}$ Cycle Time	8 × T24	16 × T24
T30 = $\overline{TCLK}$ High Time	T25 (Typically)	T25 (Typically)
T31 = $\overline{TCLK}$ Low Time	7 × T24 + T26 (Typically)	15 × T24 + T26 (Typically)

*A non-symmetrical clock should be provided so that T25 is less than 1000 ns.
T24 = Serial Clock Period
T25 = Serial Clock High Time
T26 = Serial Clock Low Time

High Speed Mode

- Applies for $\overline{TxC}$, $\overline{RxC}$
- f max = 5 MHz ±100 ppm
- For Manchester, symmetry is required: $T_{63}, T_{64} = \frac{1}{2f} \pm 5\%$

High Integration Mode

Symbol	Parameter	Min	Max	Units	Test Conditions
External (Fast) Clock Parameters					
T24	Fast Clock Cycle	62.5		ns	*1
T25	$\overline{TFC}$ High Time	18.5	1000	ns	*1, *14
T26	$\overline{TFC}$ Low Time	23.5		ns	*1
T27	$\overline{TFC}$ Rise Time		5	ns	*1
T28	$\overline{TFC}$ Fall Time		5	ns	*1
Transmit Clock Parameters					
T29	Transmit Clock Cycle	500		ns	*3, *12
T30	$\overline{TCLK}$ High Time	*8	1070	ns	*3
T31	$\overline{TCLK}$ Low Time	*9			*3
T32	$\overline{TCLK}$ Rise Time		15	ns	*3
T33	$\overline{TCLK}$ Fall Time		15	ns	*3

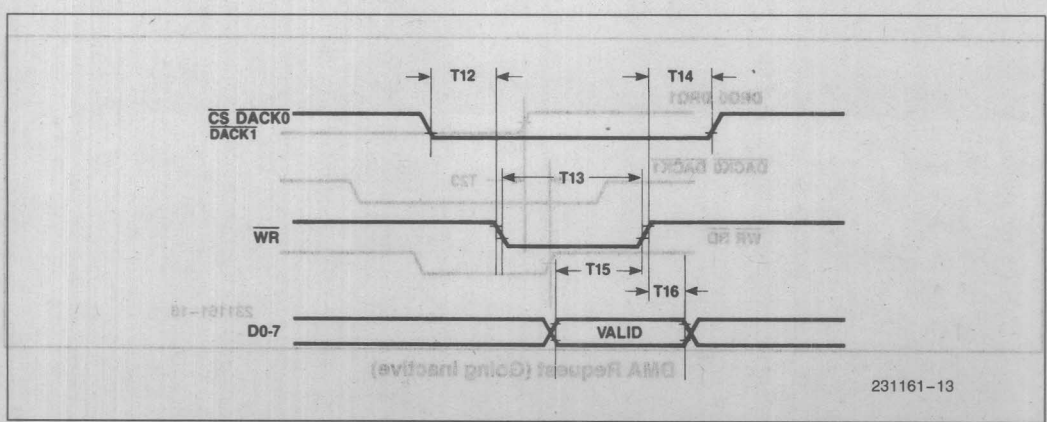
High Integration Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Transmit Data Parameters (Manchester, Differential Manchester)					
T34	TxD Transition-Transition	4T24-10		ns	*12
T35	$\overline{TCLK}$ Low to TxD Transition Half Bit Cell		*10		*2, *12
T36	$\overline{TCLK}$ Low to TxD Transition Full Bit Cell		*11		*2, *12
T37	TxD Rise Time		15	ns	*2
T38	TxD Fall Time		15	ns	*2
Transmit Data Parameters (NRZI)					
T39	TxD Transition-Transition	8T24-10		ns	*12
T40	$\overline{TCLK}$ Low to TxD Transition		*10		*2, *12
T41	$\overline{TxD}$ Rise Time		15	ns	*2
T42	TxD Fall Time		15	ns	*2
$\overline{RTS}$, $\overline{CTS}$, Parameters					
T43	$\overline{TCLK}$ Low To $\overline{RTS}$ Low		*10		*3, *12
T44	$\overline{CTS}$ Low to $\overline{TCLK}$ Low $\overline{CTS}$ Setup Time	65		ns	
T45	$\overline{TCLK}$ low to $\overline{RTS}$ High		*10		*3, *12
T46	$\overline{TCLK}$ Low to $\overline{CTS}$ Invalid. $\overline{CTS}$ Hold Time	20		ns	*4, *13
T47	$\overline{CTS}$ High to $\overline{TCLK}$ Low. $\overline{CTS}$ Setup Time to Stop Transmission	65		ns	*4
IFS Parameters					
T48	Interframe Delay	*5			
Collision Detect Parameter					
T49	$\overline{CDT}$ Low to $\overline{TCLK}$ High. External Collision Detect Setup Time	50		ns	*13
T50	$\overline{CDT}$ High to $\overline{TCLK}$ Low	50		ns	*13
T51	$\overline{TCLK}$ High to $\overline{CDT}$ Inactive. $\overline{CDT}$ Hold Time	20		ns	*13

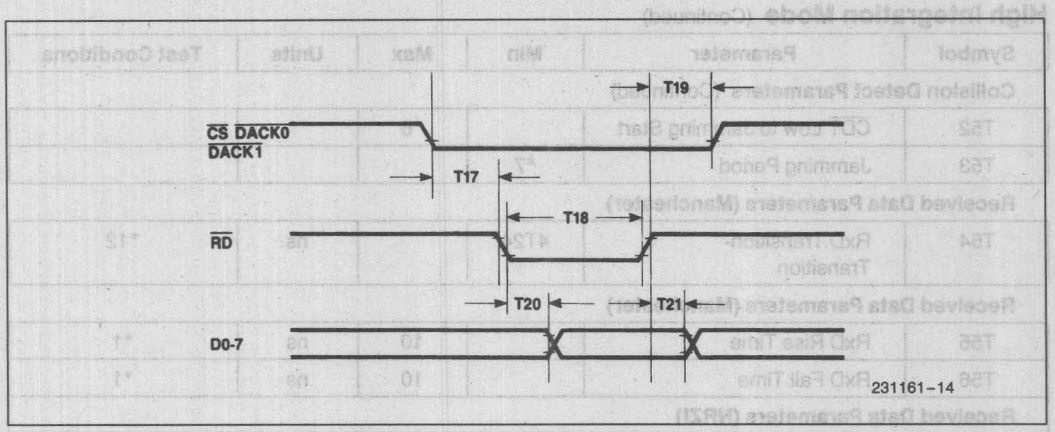
High Integration Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
Collision Detect Parameters (Continued)					
T52	$\overline{CDT}$ Low to Jamming Start		*6		
T53	Jamming Period	*7			
Received Data Parameters (Manchester)					
T54	RxD Transition-Transition	4T24		ns	*12
Received Data Parameters (Manchester)					
T55	RxD Rise Time		10	ns	*1
T56	RxD Fall Time		10	ns	*1
Received Data Parameters (NRZI)					
T57	RxD Transition-Transition	8T24		ns	*12
T58	RxD Rise Time		10	ns	*1
T59	RxD Fall Time		10	ns	*1

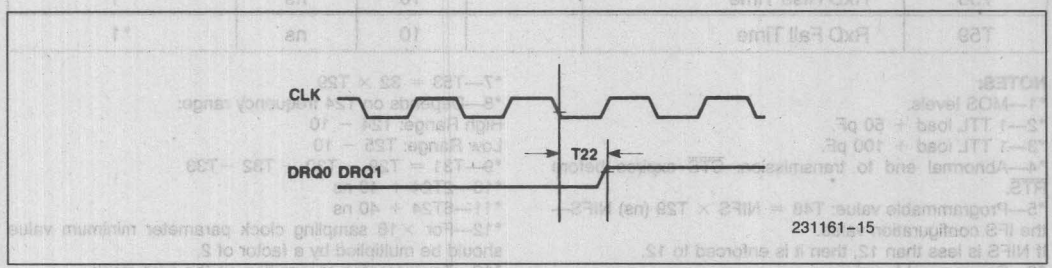
NOTES:
*1—MOS levels.
*2—1 TTL load + 50 pF.
*3—1 TTL load + 100 pF.
*4—Abnormal end to transmission: $\overline{CTS}$ expires before $\overline{RTS}$.
*5—Programmable value: T48 = NIFS × T29 (ns) NIFS—the IFS configuration value.
If NIFS is less than 12, then it is enforced to 12.
*6—Programmable value:
T52 = NCDF × T29 + (12 to 15) × T29 (if collision occurs after preamble).
*7—T53 = 32 × T29
*8—Depends on T24 frequency range:
High Range: T24 − 10
Low Range: T25 − 10
*9—T31 = T29 − T30 − T32 −T33
*10—2T24 + 40 ns
*11—6T24 + 40 ns
*12—For ×16 sampling clock parameter minimum value should be multiplied by a factor of 2.
*13—To guarantee recognition on the next clock.
*14—62.5 ns minimum in Low Range.



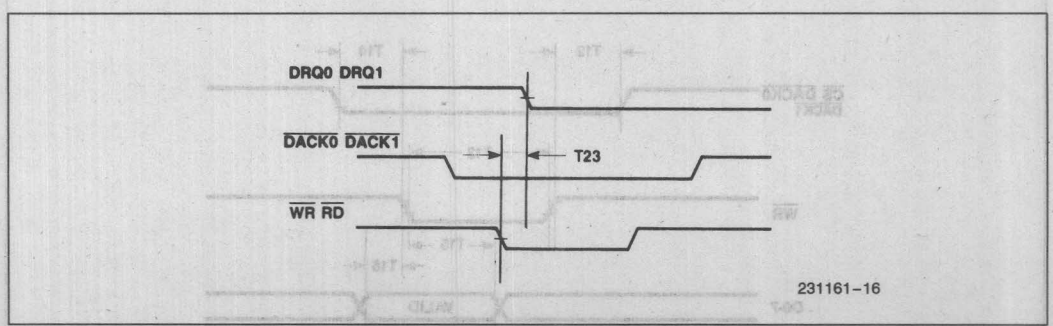
Write Timing



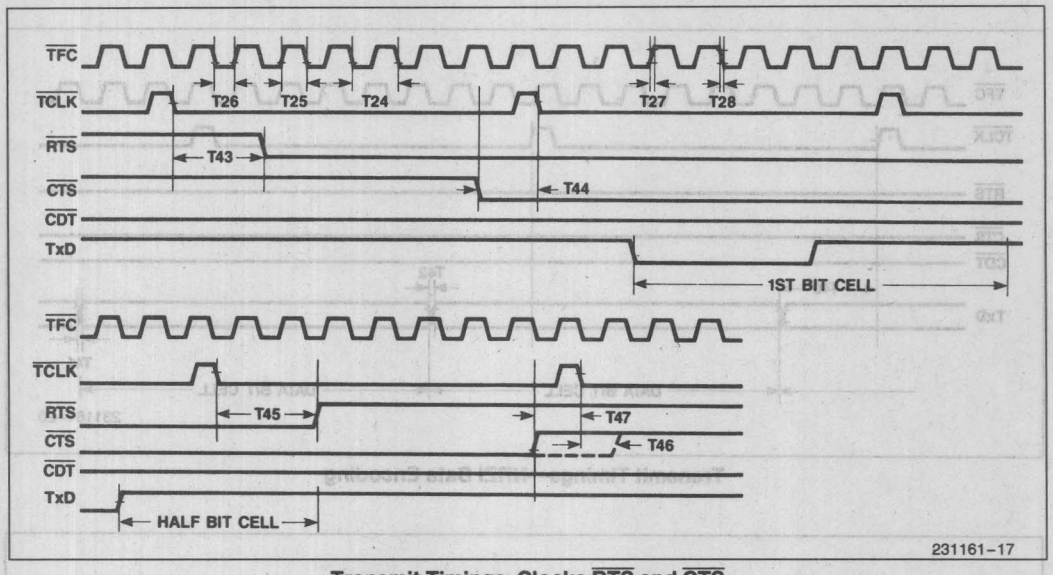
Read Timing



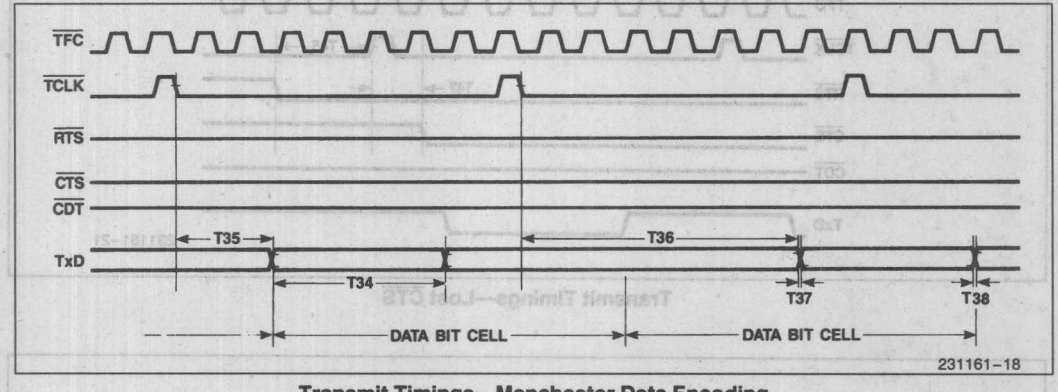
DMA Request (Going Active)



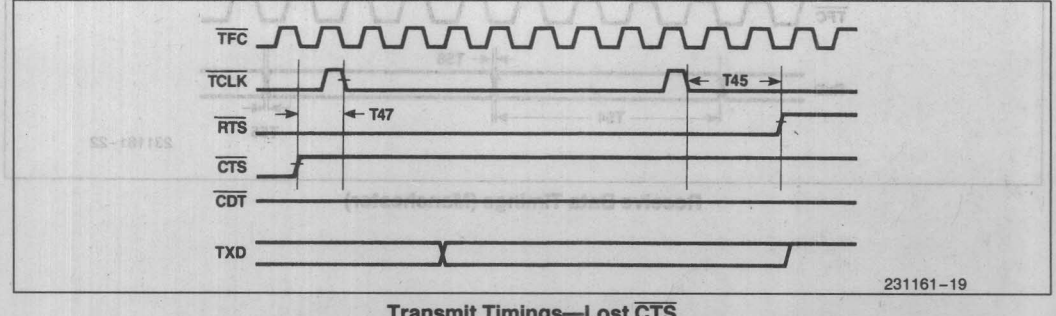
DMA Request (Going Inactive)



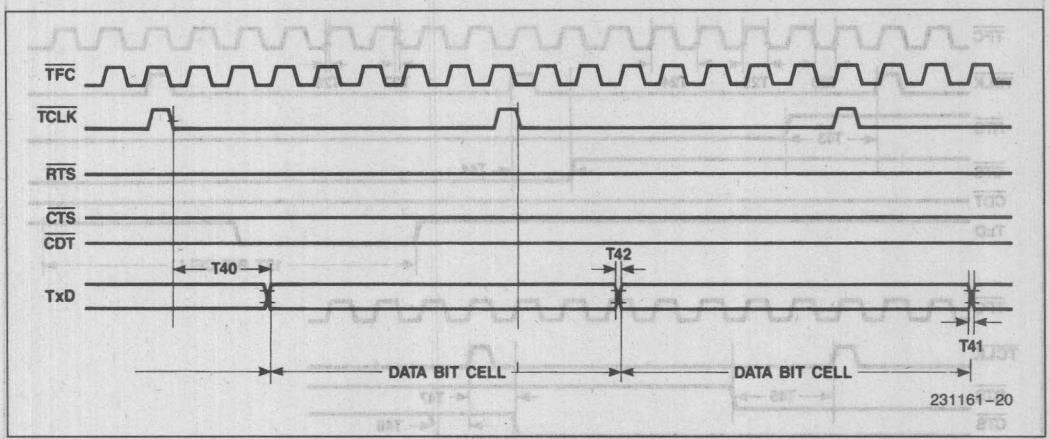
Transmit Timings: Clocks $\overline{RTS}$ and $\overline{CTS}$



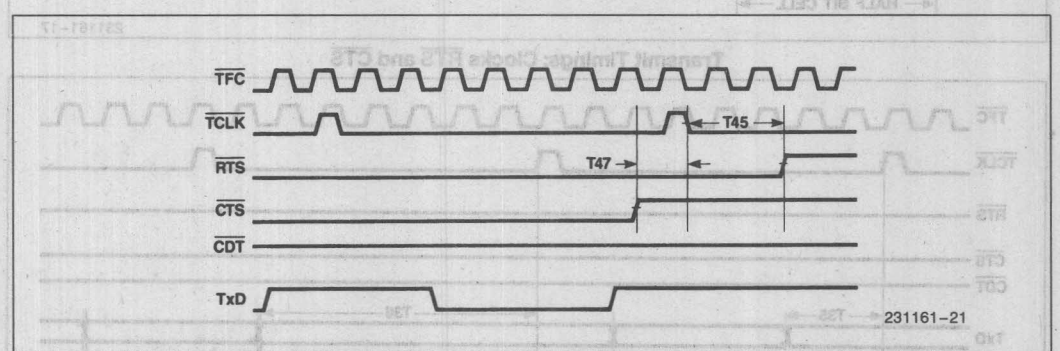
Transmit Timings—Manchester Data Encoding



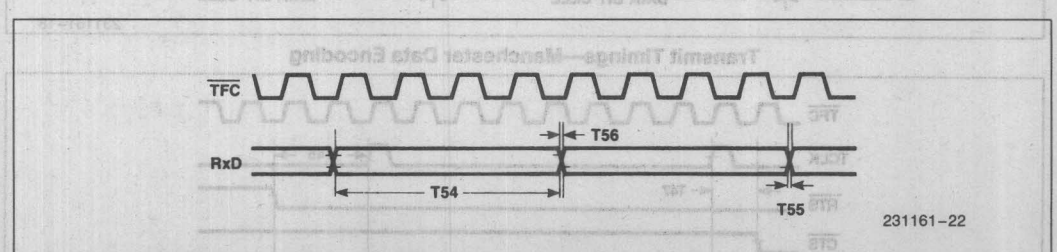
Transmit Timings—Lost $\overline{CTS}$



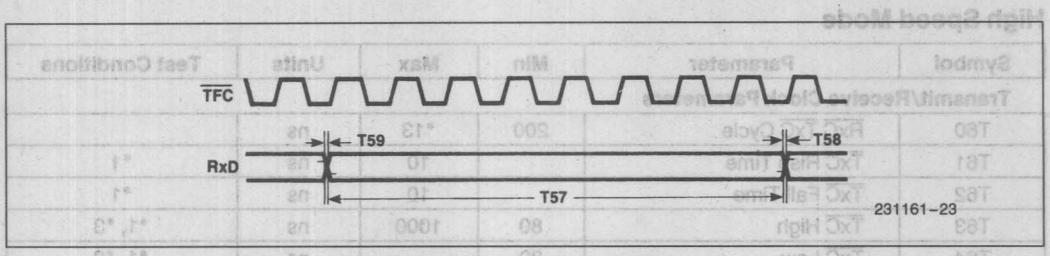
Transmit Timings—NRZI Data Encoding



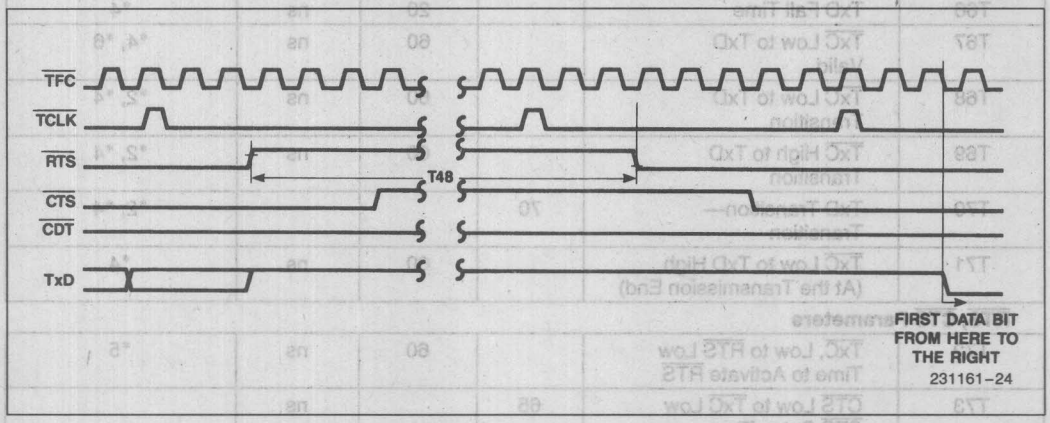
Transmit Timings—Lost $\overline{CTS}$



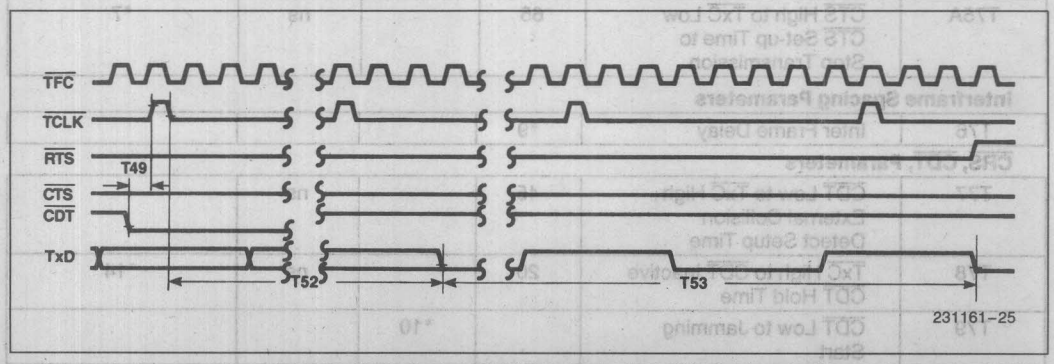
Receive Data Timings (Manchester)



Receive Data Timings (NRZI)



Transmit Timings—Interframe Spacing



Transmit Timings—Collision Detect and Jamming

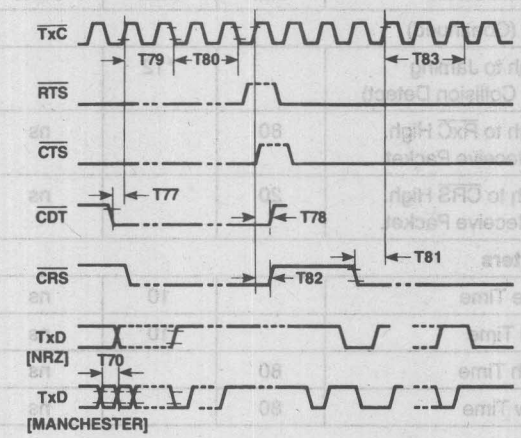
High Speed Mode

Symbol	Parameter	Min	Max	Units	Test Conditions
Transmit/Receive Clock Parameters					
T60	$\overline{RxC}$ $\overline{TxC}$ Cycle	200	*13	ns	
T61	$\overline{TxC}$ Rise Time		10	ns	*1
T62	$\overline{TxC}$ Fall Time		10	ns	*1
T63	$\overline{TxC}$ High	80	1000	ns	*1, *3
T64	$\overline{TxC}$ Low	80		ns	*1, *3
Transmit Data Parameters					
T65	TxD Rise Time		20	ns	*4
T66	TxD Fall Time		20	ns	*4
T67	$\overline{TxC}$ Low to TxD Valid		60	ns	*4, *6
T68	$\overline{TxC}$ Low to TxD Transition		60	ns	*2, *4
T69	$\overline{TxC}$ High to TxD Transition		60	ns	*2, *4
T70	TxD Transition— Transition	70			*2, *4
T71	$\overline{TxC}$ Low to TxD High (At the Transmission End)		60	ns	*4
$\overline{RTS}$, $\overline{CTS}$ Parameters					
T72	$\overline{TxC}$, Low to $\overline{RTS}$ Low Time to Activate $\overline{RTS}$		60	ns	*5
T73	$\overline{CTS}$ Low to $\overline{TxC}$ Low $\overline{CTS}$ Setup Time	65		ns	
T74	$\overline{TxC}$ Low to $\overline{RTS}$ High		60	ns	*5
T75	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid	20		ns	
T75A	$\overline{CTS}$ High to $\overline{TxC}$ Low $\overline{CTS}$ Set-up Time to Stop Transmission	65		ns	*7
Interframe Spacing Parameters					
T76	Inter Frame Delay	*9			
$\overline{CRS}$, $\overline{CDT}$, Parameters					
T77	$\overline{CDT}$ Low to $\overline{TxC}$ High External Collision Detect Setup Time	45		ns	
T78	$\overline{TxC}$ High to $\overline{CDT}$ Inactive $\overline{CDT}$ Hold Time	20		ns	*14
T79	$\overline{CDT}$ Low to Jamming Start		*10		
T80	Jamming Period	*11			
T81	$\overline{CRS}$ Low to $\overline{TxC}$ High Carrier Sense Setup Time	45		ns	*14
T82	$\overline{TxC}$ High to $\overline{CRS}$ Inactive $\overline{CRS}$ Hold Time	20		ns	*14

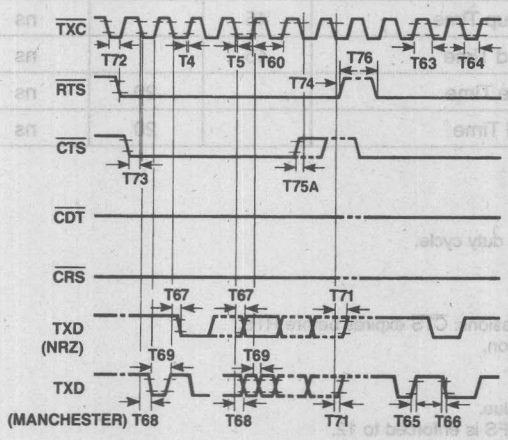
High Speed Mode (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
	$\overline{CRS}$, $\overline{CDT}$, Parameters (Continued)				
T83	$\overline{CRS}$ High to Jaming (Internal Collision Detect)		*12		
T84	$\overline{CRS}$ High to $\overline{RxC}$ High. End of Receive Packet	80		ns	
T85	$\overline{RxC}$ High to $\overline{CRS}$ High. End of Receive Packet.	20		ns	
	Receive Clock Parameters				
T86	$\overline{RxC}$ Rise Time		10	ns	*1
T87	$\overline{RxC}$ Fall Time		10	ns	*1
T88	$\overline{RxC}$ High Time	80		ns	*1
T89	$\overline{RxC}$ Low Time	80		ns	*1
	Received Data Parameters				
T90	RxD Setup Time	45		ns	*1
T91	RxD Hold Time	45		ns	*1
T92	RxD Rise Time		20	ns	*1
T93	RxD Fall Time		20	ns	*1

NOTES:
*1 — MOS levels.
*2 — Manchester only.
*3 — Manchester. Needs 50% duty cycle.
*4 — 1 TTL load + 50 pF.
*5 — 1 TTL load + 100 pF.
*6 — NRZ only.
*7 — Abnormal end to transmissions: $\overline{CTS}$ expires before $\overline{RTS}$.
*8 — Normal end to transmission.
*9 — Programmable value.
T76 = NIFS × T60 (ns)
NIFS - the IFS configuration value.
If NIFS is less than 12, then NIFS is enforced to 12.
*10 — Programmable value:
T79 = NCDF × T60 + (12 to 15) × T60 (ns) (if collision occurs after preamble).
*11 — T80 = 32 × T60
*12 — Programmable value:
NCSF × TTRC + (12 to 15) × TTRC
T83 = NCSF × T60 + (12 to 15) × T60
NCDF - collision detect filter configuration value.
*13 — 2000 ns if configured for Manchester encoding.
*14 — To guarantee recognition on the next clock.

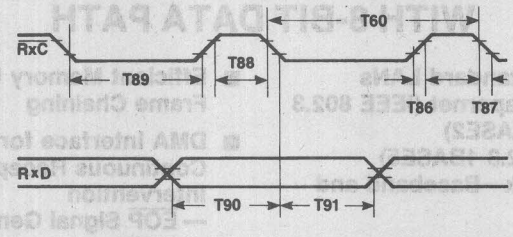


231161-26

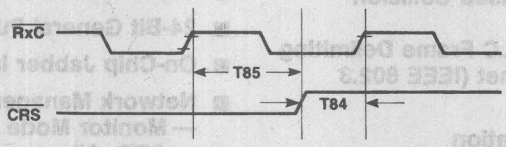


231161-27

Transmit Data Waveforms



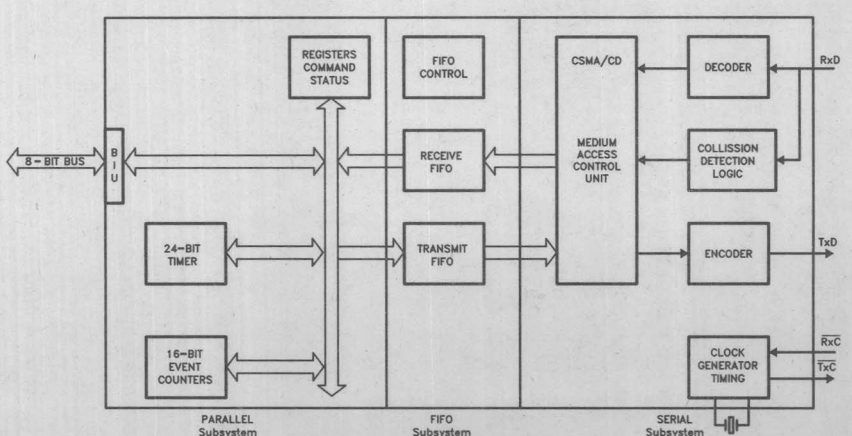
Receive Data Waveforms (NRZ)



Receive Data Waveforms

82590
ADVANCED CSMA/CD LAN CONTROLLER WITH 8-BIT DATA PATH

- Supports Industry Standard LANs
  - Ethernet and Cheapernet (IEEE 802.3 10BASE5 and 10BASE2)
  - StarLAN (IEEE 802.3 1BASE5)
  - IBM™ PC Network—Baseband and Broadband
- Integrates Physical and Data Link Layers of OSI Model
  - Complete CSMA/CD Medium Access Control (MAC) Functions
  - Manchester, Differential Manchester, or NRZI Encoding/Decoding
  - On-Chip, Logic-Based Collision Detection
  - IEEE 802.3 or HDLC Frame Delimiting
  - Broadband Ethernet (IEEE 802.3 10BROAD36)
- Two Modes of Operation
  - Bit Rates up to 4 Mb/s with On-Chip Encoder/Decoder (High-Integration Mode)
  - Bit Rates up to 20 Mb/s with External Encoder/Decoder (High-Speed Mode)
- High-Performance System Interface
  - 16-MHz Clock, 2 Clocks per Transfer
  - 64 Bytes of Configurable FIFO
- Efficient Memory Use via Buffer and Frame Chaining
- DMA Interface for Retransmission and Continuous Reception without CPU Intervention
  - $\overline{EOP}$ Signal Generation for 8237 and 82380
  - Tightly Coupled Interface to 82560 Host Interface and Memory Manager
- 82588 Pin- and Software-Compatible Mode
- Local and Remote Power-Down Modes
- 24-Bit General Purpose Timer
- On-Chip Jabber Inhibit Function
- Network Management and Diagnostics
  - Monitor Mode
  - CRC, Alignment, and Short Frame Error Detection
  - Three 16-Bit Event Counters
  - Short or Open Circuit Localization
  - Self-Test Diagnostics
  - Internal and External Loopback Operation
  - Internal Register Dump
- High-Speed CHMOS III Technology



290147-1

Figure 1. 82590 Block Diagram

*IBM, PC, PCAT, PCXT are trademarks of International Business Machines.

October 1990
Order Number: 290147-004

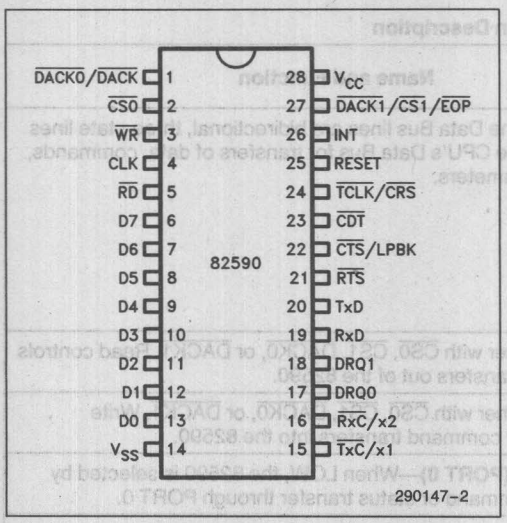


Figure 2. 82590 Pin Configuration (DIP)

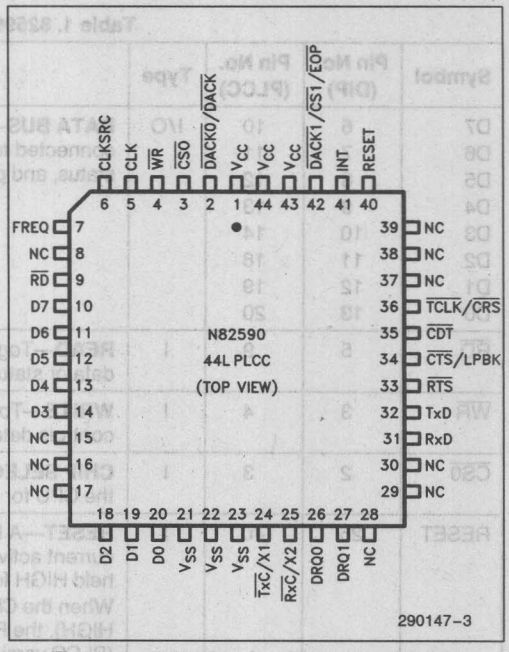


Figure 3. 82590 Pin Configuration (PLCC)

The 82590 is a second-generation, 8-bit data path CSMA/CD controller. Its system interface enables efficient operation with a wide variety of Intel microprocessors (such as iAPX 188, 186, 286, or 386) and industry standard buses (such as the IBM PC I/O channel or Personal System/2™ Micro Channel™). The 82590 can be configured to support a wide variety of industry standard networks, including StarLAN and Ethernet/Cheapernet.

The 82590 provides a natural upgrade path for existing 82588 applications, since it is pin and software compatible with its predecessor. Its rich incremental functionality compared to the 82588 can be utilized by selectively modifying existing software drivers.

Together with the 82560 (Host Interface and Memory Manager) the 82590 offers a complete solution for CSMA/CD LAN adapters oriented to the IBM PC environment. The 82590 fully conforms to existing IEEE 802.3 standards (1BASE5, 10BASE5, 10BASE2, and 10BROAD36). Intel also offers the 82592, a 16-bit data path version of the 82590, for higher performance applications.

The 82590 is available in a 28-pin Plastic DIP or a 44-pin PLCC package. It is fabricated with Intel's reliable CHMOS III technology.

Table 1. 82590 Pin Description

Symbol	Pin No. (DIP)	Pin No. (PLCC)	Type	Name and Function
D7 D6 D5 D4 D3 D2 D1 D0	6 7 8 9 10 11 12 13	10 11 12 13 14 18 19 20	I/O	**DATA BUS**—The Data Bus lines are bidirectional, three-state lines connected to the CPU's Data Bus for transfers of data, commands, status, and parameters.
$\overline{RD}$	5	9	I	**READ**—Together with $\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$, Read controls data or status transfers out of the 82590.
$\overline{WR}$	3	4	I	**WRITE**—Together with $\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$, Write controls data or command transfers into the 82590.
$\overline{CS0}$	2	3	I	**CHIP SELECT (PORT 0)**—When LOW, the 82590 is selected by the CPU for command or status transfer through PORT 0.
RESET	25	40	I	**RESET**—A HIGH signal on this pin causes the 82590 to terminate current activity. This signal is internally synchronized and must be held HIGH for at least four Clock (CLK) cycles. When the Clock signal is provided internally (CLKSRC is strapped HIGH), the RESET signal must be held HIGH for at least 50 μs. (PLCC version only.)
INT	26	41	O	**INTERRUPT**—A HIGH signal on this pin notifies the CPU that the 82590 is requesting an interrupt.
DRQ0	17	26	O	**DMA REQUEST (CHANNEL 0)**—This pin is used by the 82590 to request DMA transfer. DRQ0 remains HIGH as long as the 82590 requires DMA transfers. Burst transfers are thus possible. When the 82590 is programmed for Tightly Coupled Interface, the 82590 notifies the DMA controller of the status of transmission or reception, using this pin together with $\overline{EOP}$.
DRQ1	18	27	O	**DMA REQUEST (CHANNEL 1)**—This pin is used by the 82590 to request DMA transfer. DRQ1 remains HIGH as long as the 82590 requires DMA transfers. Burst transfers are thus possible. When the 82590 is programmed for Tightly Coupled Interface, the 82590 notifies the DMA controller of the status of transmission or reception, using this pin together with $\overline{EOP}$.
$\overline{DACK0}$ $\overline{DACK}$	1	2	I	**DMA ACKNOWLEDGE (CHANNEL 0)**—When LOW, this input signal from the DMA controller notifies the 82590 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 0. **DMA ACKNOWLEDGE (CHANNELS 0 AND 1)**—When the $\overline{DACK1/CS1/EOP}$ pin is programmed to $\overline{CS1/EOP}$, this pin provides a DMA acknowledge for both channels 0 and 1. Two DMA acknowledge signals from the DMA controller, $\overline{DACK0}$ and $\overline{DACK1}$, must be externally ANDed in this mode of operation.

Table 1. 82590 Pin Description (Continued)

Symbol	Pin No. (DIP)	Pin No. (PLCC)	Type	Name and Function
$\overline{DACK1}$ / $\overline{CS1}/\overline{EOP}$	27	42	I	This is a multifunction, bidirectional pin which can be programmed to $\overline{DACK1}$ or $\overline{CS1}/\overline{EOP}$ during configuration. When it is configured for $\overline{EOP}$, it provides an open-drain output. **DMA ACKNOWLEDGE (CHANNEL 1)**—When LOW, this input signal from the DMA controller notifies the 82590 that the requested DMA cycle is in progress. This signal acts similarly to Chip Select for data and parameter transfers, using DMA channel 1. **CHIP SELECT (PORT 1)**—When LOW, the 82590 is selected by the CPU for command or status transfer through PORT 1. **END OF PROCESS**—A LOW output signal requests the DMA controller to terminate the active DMA service.
			I/O	
CLK	4	5	I	**CLOCK**—In the 28-pin DIP, this is only an input pin. A TTL-compatible clock input to this pin provides the timing for the 82590 parallel subsystem. In the 44-pin PLCC, this pin can be a clock input or output, depending on the state of CLKSRC. If CLKSRC is strapped LOW, this pin is a clock input which provides timing for the 82590 parallel subsystem. If CLKSRC is strapped HIGH, the clock for the 82590 parallel subsystem is generated from the internal clock generator. The CLK pin is then a clock output and provides a clock signal whose frequency can be one-half of or identical to, the frequency of the internally generated parallel subsystem clock, depending on the state of FREQ. Note that the maximum frequency of the clock signal supplied by the CLK pin is 8 MHz.
			I/O	

CLKSRC	FREQ	CLK		Clock for the Parallel Subsystem
		Type	Signal	
0 (LOW)	Don't Care	I	Clock	Clock as Provided on the CLK Pin
1 (HIGH)	1	O	Internal Parallel Subsystem Clock Divided by Two	Prescaled Clock Generated from the Internal Clock Generator
1	0	O	Internal Parallel Subsystem Clock	Prescaled Clock Generated from the Internal Clock Generator

CLKSRC	NA	6	I	**CLOCK SOURCE**—When strapped LOW, a clock signal on the CLK pin provides timing for the parallel subsystem. When strapped HIGH, timing for the parallel subsystem is internally generated from the clock generator provided in the serial subsystem. The internal prescaler is programmed during configuration to determine the frequency of the clock for the parallel subsystem.
FREQ	NA	7	I	**FREQUENCY**—When strapped LOW, CLK has an output frequency equal to that of the internal parallel subsystem clock. When strapped HIGH, CLK has an output frequency one-half that of the internal parallel subsystem clock. The state of this pin is relevant only when CLKSRC is strapped HIGH.

Table 1. 82590 Pin Description (Continued)

Symbol	Pin No. (DIP)	Pin No. (PLCC)	Type	Name and Function
				High Integration Mode
X1/X2	15/16	24/25	I	**OSCILLATOR INPUTS**—These inputs may be used to connect a quartz crystal which controls the internal clock generator for the serial subsystem. When CLKSRC is strapped HIGH, the clock generator also provides a clock for the parallel subsystem. X1 may also be driven by a MOS-level clock whose frequency is 8, 10, 16, or 18 times the bit rate of Transmit/Receive data. X2 must be left floating if X1 is connected to an external MOS clock.
				High Speed Mode
$\overline{TxC}$	15	24	I	**TRANSMIT CLOCK**—This signal provides the fundamental timing for the serial subsystem. The clock is also used to transmit data synchronously on the TxD pin. For NRZ encoding, data is transferred to the TxD pin on the HIGH to LOW clock transition. For Manchester encoding, the transmitted bit center is aligned with the LOW to HIGH transition.
$\overline{RxC}$	16	25	I	**RECEIVE CLOCK**—This clock is used to synchronously sample data on the RxD pin. Only NRZ data format is supported for reception. The state of the RxD pin is sampled on the HIGH to LOW transition.
$\overline{TCLK}/\overline{CRS}$	24	36	I	**CARRIER SENSE**—In High Speed Mode this pin is Carrier Sense, $\overline{CRS}$, and is used to notify the 82590 that the serial link is active
			O	**TRANSMIT CLOCK**—In High Integration Mode this pin is Transmit Clock, $\overline{TCLK}$.
$\overline{CDT}$	23	35	I	**COLLISION DETECT**—This input notifies the 82590 that a collision has occurred. In High Speed Mode a collision is sensed by this pin only when the 82590 is configured for external Collision Detect (external means are then required for collision detection). In High Integration Mode collisions are sensed by this pin regardless of the internal or external Collision Detect configuration of the 82590.
RxD	19	31	I	**RECEIVE DATA**—This pin receives serial data. It must be HIGH when not receiving.
TxD	20	32	O	**TRANSMIT DATA**—This pin transmits data to the serial link. It is HIGH when not transmitting.
$\overline{RTS}$	21	33	O	**REQUEST TO SEND**—When this signal is LOW the 82590 notifies the channel that it has data to transmit. It is forced HIGH after a reset or when transmission is stopped.
$\overline{CTS}$/LPBK	22	34	I/O	**CLEAR TO SEND**—An active LOW signal which enables the 82590 to start transmitting data. Asserting this signal HIGH stops the transmission. **LOOPBACK**—This pin, in conjunction with a pull-down resistor, can be programmed to provide an active HIGH loopback signal to the external interface device.
V_{CC}	28	1 43 44		**POWER:** +5V ±10%
V_{SS}	14	21 22 23		**GROUND:** 0V

FUNCTIONAL DESCRIPTION

Internal Architecture

The 82590 consists of a parallel subsystem, a serial subsystem, and a FIFO subsystem (see Figure 1).

Parallel Subsystem

The parallel subsystem consists of a bus interface unit (BIU), command and status registers, a 24-bit general purpose timer, and three 16-bit event counters.

The BIU provides an 8-bit data bus interface to the external system bus. It handles all data transfers to and from memory (at speeds up to 8 Mbytes/sec.), accepts commands from the CPU, and provides status to the CPU. There are two separate I/O ports, Port 0 and Port 1; and two separate DMA channels, Channel 0 and Channel 1. Port 0 is the 82588-compatible I/O port through which the CPU issues commands such as Transmit and Receive Enable. The 82590's enhanced features, such as the general purpose timer and event counters, are accessed through Port 1. The two DMA channels are independent of each other and can be used for high-performance operations such as simultaneous transmission and reception.

The 24-bit timer consists of a 24-bit maximum count register, a 24-bit count register, and associated control bits in the command registers. Its clock source can be the transmit clock or the parallel subsystem clock. The timer can be programmed to halt or continue on a terminal count with or without causing an interrupt.

The three 16-bit event counters can be programmed to count valid frames, collided frames, and errored (CRC or Alignment) frames. When these event counters are used in Monitor mode, the 82590 is capable of maintaining the network statistics by itself; i.e., without requesting DMA services or causing interrupts to the CPU.

Serial Subsystem

The serial subsystem consists of a CSMA/CD unit, a data encoder and decoder, collision detect and carrier sense logic, and a clock generator.

The 82590's CSMA/CD unit is highly flexible in implementing the CSMA/CD protocol. It can operate in a variety of IEEE 802.3 and other CSMA/CD LAN environments, including 1BASE5 (StarLAN, 10BASE5 (Ethernet), 10BASE2 (Cheapernet), and the IBM™ PC Network (Baseband and Broadband). The programmable parameters include:

- Framing (IEEE 802.3 Framing or HDLC Framing)
- Address Field Length
- Station Priority
- Interframe Spacing
- Slot Time
- CRC-32 or CRC-16

The encoder and decoder in the serial subsystem is capable of NRZI, Manchester, and Differential Manchester encoding and decoding at bit rates up to 4 Mb/s in High-Integration Mode, and Manchester encoding at bit rates up to 20 Mb/s in High-Speed Mode. A digital phase-lock loop is used in High-Integration Mode to decode the receive data and to generate the synchronous receive clock.

The collision detect and carrier sense logic generate the internal collision detect and carrier sense signals for the CSMA/CD unit.

The 82590 implements several different internal, logic-based collision detect mechanisms. Two of these, Code Violation and Bit Comparison, are also available with the 82588. The Code Violation method defines a collision where a transition edge occurs outside the area of normal transitions (as specified by the data encoding method). For example, if there are no mid-bit cell transitions in the Manchester encoded data, this method interprets that condition as a collision. The Bit Comparison method compares the signature of the transmitted frame to the signature of the received frame. If the signatures are different, a collision is assumed to have occurred. Two other internal collision detect methods implemented in the 82590 are Source Address Comparison and StarLAN CPS (Collision Presence Signal) Recognition. The Source Address Comparison compares the source address field of the transmitted frame to the source address field of the received frame. If the source addresses are different, it assumes that a collision has occurred resulting in data corruption in the source address field. The StarLAN CPS Recognition method looks for the specific collision presence signal defined by the IEEE 802.3 1BASE5 standard. Other abnormal circumstances, such as no carrier for more than one-half slot time in the receive channel during transmission, are interpreted as collisions by the 82590.

In addition to these internal, logic-based collision detection methods, an external means of collision detection can be used in parallel by using the $\overline{CDT}$ input pin.

The clock generator in the serial subsystem is available only in High-Integration Mode and provides timing for the serial subsystem. The clock signal can also be routed to the parallel subsystem, if so desired. The oscillator circuit is designed for use with an external, parallel resonant, fundamental mode crystal. The crystal frequency should be selected at 8×, 10×, 16×, or 18× the required serial bit rate.

FIFO Subsystem

The FIFO subsystem is located between the parallel subsystem and the serial subsystem. It consists of a transmit FIFO, a receive FIFO, and FIFO control logic. The transmit and receive FIFOs are independent of each other and individually provide optimal interfaces between the two subsystems which may have different speeds. There is a total of 64 bytes that can be used for the two separate FIFOs. During configuration these 64 bytes can be divided into one of four possible combinations: 16 and 16 bytes, 16 and 48 bytes, 32 and 32 bytes, or 48 and 16 bytes for the transmit and receive FIFO respectively. The FIFO threshold is also programmed during configuration.

PROGRAMMING MODEL—REGISTER OVERVIEW

Figure 4 shows the 82590 internal registers that are directly accessible through the 8-bit I/O ports: Port 0 and Port 1. The registers enclosed in darker lines are 82588-compatible registers and are accessible only through Port 0.

Figure 5 shows the Port 0 commands. All of the Port 0 commands are compatible with the 82588 except for the NOP command with the channel bit set to 1. If the NOP command is executed with the channel bit set to 1, the active port is switched to Port 1. Port 0, which is selected by $\overline{CS0}$ in hardware, logically becomes Port 1. When the hardware does not support the second chip select, $\overline{CS1}$, this software port switch command is used. Figure 6 shows the Port 1 commands. When the SWT-TO-PORT-0 command is executed, the active port is switched back to Port 0.

The 82590 can be configured to have 4 or 6 bytes of status registers in Port 0 (see Figures 4 and 7). When configured to 4 bytes of status registers, formats of these registers are identical to those of the 82588. The first three status registers (STATUS 0 through 2) contain the information about the last command executed or the last frame received. The last status register, STATUS 3, contains the state of the 82590. When the 82590 is configured to 6 bytes of status registers, the two additional bytes are used to report a more complete status of the most recently received frame.

Status of the timer and event counters is available in the Port 1 status registers as shown in Figure 8.

82590 AND HOST INTERACTION

The CPU interacts with the 82590 through the system's memory and the 82590's on-chip registers. The CPU creates a data structure in memory, programs the external DMA controller with the start address and byte count of the memory block, and issues a command to the 82590.

The chip select and interrupt lines are used to communicate between the 82590 and the CPU as shown in Figure 9. The interrupt signal is used by the 82590 to attract the CPU's attention. The chip select signal is used by the CPU to attract the 82590's attention. Note that the 82590 does not have any address lines.

There are two kinds of transfers over the bus: command/status and data transfers. The command/status transfers are always performed by the CPU. The data transfers are requested by the 82590, and are usually performed by a DMA controller. Table 2 shows the command/status and data transfer control signals. The CPU writes commands to the 82590 using the $\overline{CS0}$ (or $\overline{CS1}$) and $\overline{WR}$ signals, and reads status using the $\overline{CS0}$ (or $\overline{CS1}$) and $\overline{RD}$ signals. When data transfers are performed, $\overline{DACK0}$ or $\overline{DACK1}$ must be asserted by the DMA controller instead of the Chip Select.

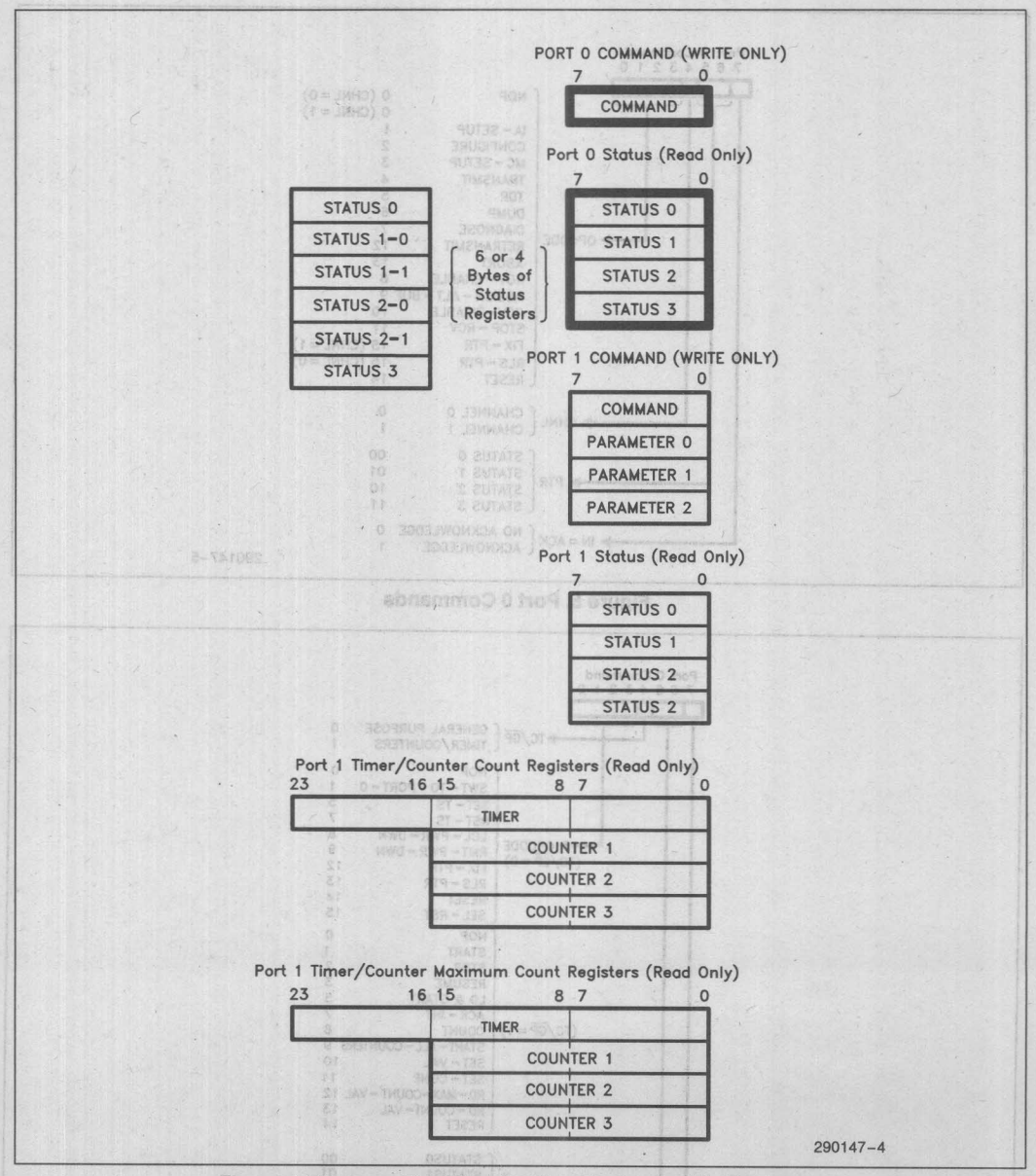


Figure 4. Programming Model—Directly Accessible Registers (Accessible Through 8-Bit I/O Port[s])

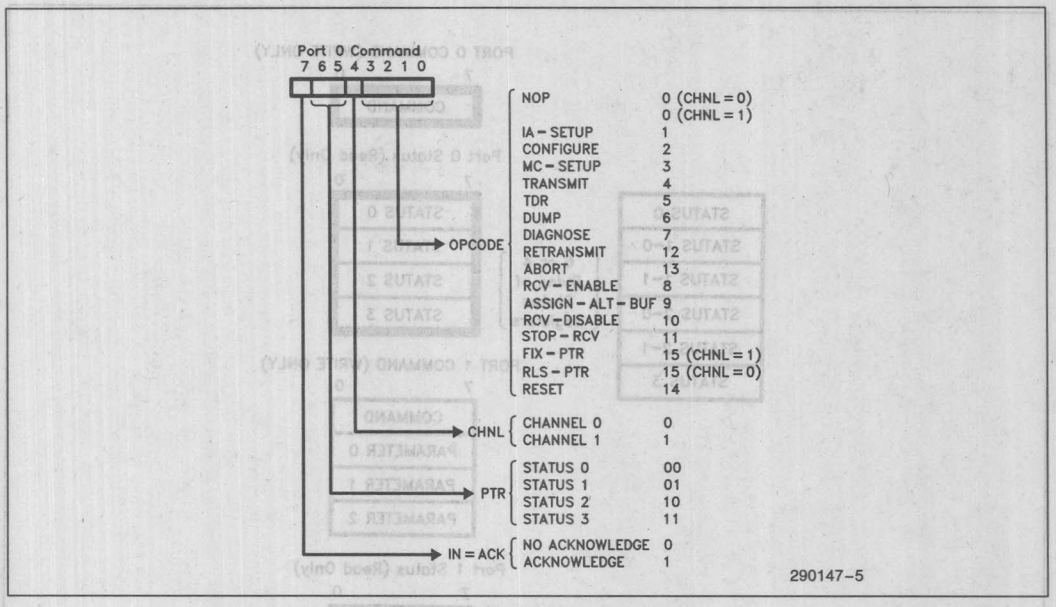


Figure 5. Port 0 Commands

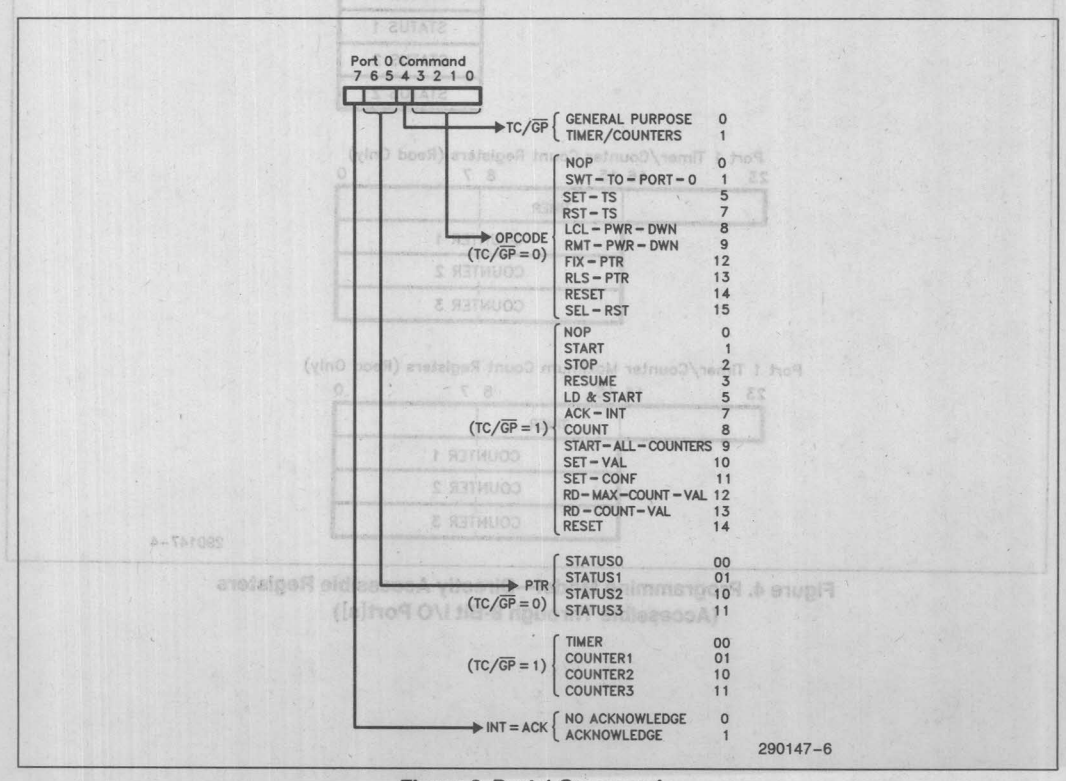
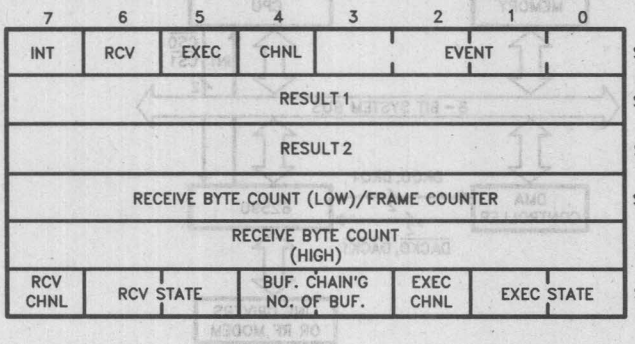


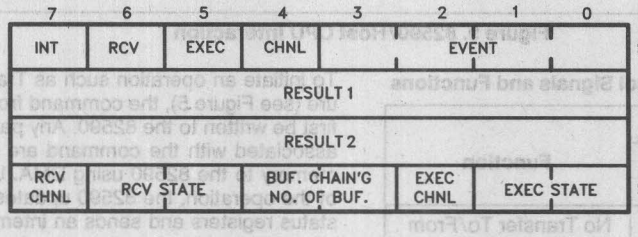
Figure 6. Port 1 Commands

Status Registers—6 Bytes



290147-7

Status Registers—4 Bytes (82588 Compatible Modes)

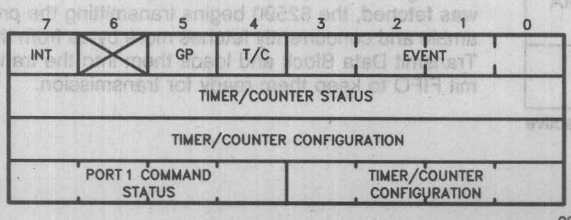


290147-8

Events	Value (Status 0)	Events	Value (Status 0)
CMOS*	0 (CHNL = 1)	Diagnose-Passed	7
IA-Setup-Done	1	End-Of-Frame	8
Configure-Done	2	Request-Next-Buffer	9
MC-Setup-Done	3	Reception-Aborted	10
Transmit-Done	4	Retransmit-Done	12
TDR-Done	5	Execution-Aborted	13
Dump-Done	6	Diagnose-Failed	15

*Available only after Hardware or Software Reset

Figure 7. Port 0 Status Registers



290147-9

Timer/Counter Events (T/C = 1)	Value* (Status 0)
Timer Expired	Bit 0 = 1
Counter 1 Expired	Bit 1 = 1
Counter 2 Expired	Bit 2 = 1
Counter 3 Expired	Bit 3 = 1
General Purpose Event (GP = 1)	**Value* (Status 0)**
REM-PWR-UP	9

The 82590 may have more than one EVENT bit set by the time the CPU reads the status register.

Figure 8. Port 1 Status Registers

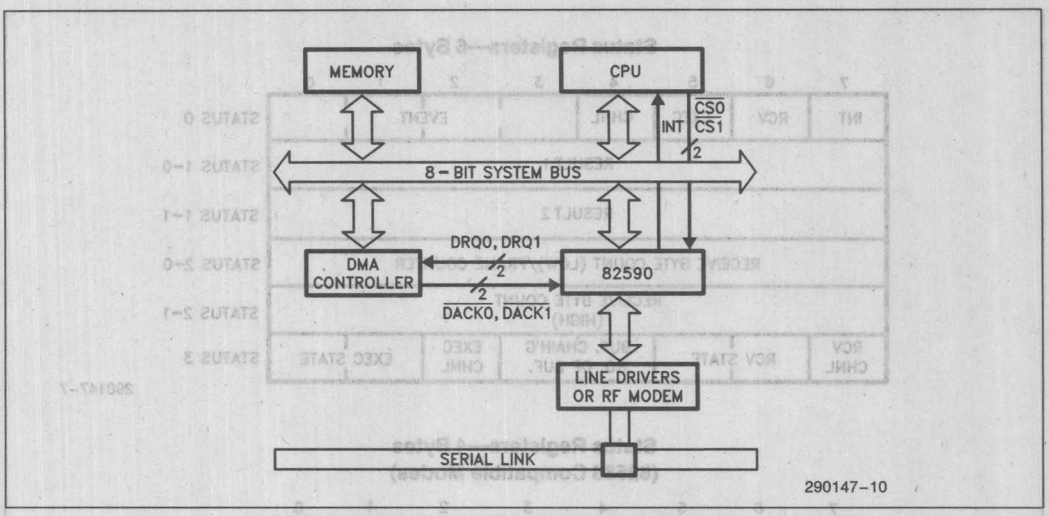


Figure 9. 82590/Host CPU Interaction

Table 2. Data Bus Control Signals and Functions

Pin Name			Function
$\overline{CS0}$ $\overline{CS1}$*	$\overline{RD}$	$\overline{WR}$	
1	X	X	No Transfer To/From Command/Status
0	1	1	
0	0	0	Illegal
0	0	1	Read from Status Register
0	1	0	Write to Command Register
$\overline{DACK0}$ $\overline{DACK1}$*	$\overline{RD}$	$\overline{WR}$	
1	X	X	No DMA Transfer
0	1	1	
0	0	0	Illegal
0	0	1	Data Read from DMA Channel 0 (or 1)
0	1	0	Data Write to DMA Channel 0 (or 1)

*Only one of $\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$ may be active at any time.

To initiate an operation such as Transmit or Configure (see Figure 5), the command from the CPU must first be written to the 82590. Any parameters or data associated with the command are transferred from memory to the 82590 using DMA. Upon completion of the operation, the 82590 updates the appropriate status registers and sends an interrupt to the CPU.

FRAME TRANSMISSION

To transmit a frame, the CPU prepares a Transmit Data Block in memory as shown in Figure 10. Its first two bytes specify the length of the rest of the block. The next few bytes (up to six) contain the destination address of the station the frame is being sent to. The rest of the block is the data field. The CPU programs the DMA controller with the start address of the block, length of the block, and other control information and then issues a Transmit command to the 82590. Upon receiving this command, the 82590 fetches the first two bytes of the block to determine its length. If the link is free and the first data byte was fetched, the 82590 begins transmitting the preamble and concurrently fetches more bytes from the Transmit Data Block and loads them into the transmit FIFO to keep them ready for transmission.

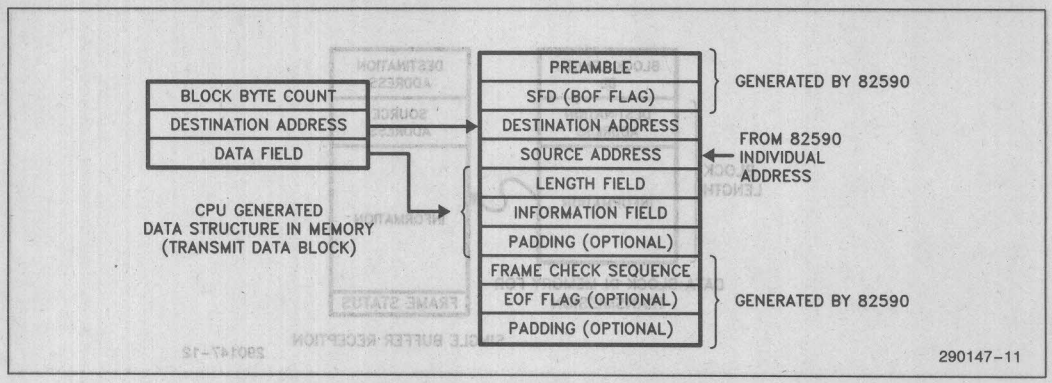


Figure 10. The 82590 Frame Structure and Location of Data Element in System Memory

The destination address is transmitted after the preamble. This is followed by the source or the station individual address, which was previously stored in the 82590 by the IA-Setup command. After this, the entire information field is transmitted, followed by a CRC field calculated by the 82590. If a collision is encountered during transmission of the frame, then the transmission is aborted after a jam pattern is sent. If the collision is detected during preamble or SFD (Start Frame Delimiter) transmission, the 82590 transmits the jam pattern after the SFD is transmitted. An interrupt is then generated to inform the CPU of the unsuccessful transmission due to a collision. The CPU reinitializes the DMA controller and issues a Retransmit command to the 82590. Retransmission is done by the CPU exactly as the Transmit command is done, except the Retransmit command keeps track of the number of collisions encountered. When the 82590 gets the Retransmit command and the backoff timer is expired, it transmits the frame again. Retransmission is repeated until the attempt is successful, or until the preprogrammed retry number expires.

If the 82590 is programmed to generate the $\overline{EOP}$ signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, retransmission is performed without CPU intervention.

FRAME RECEPTION

The 82590 can receive frames when its receiver has been enabled. The 82590 checks for an address match for an Individual address, a Multicast address, or a Broadcast address. In the Promiscuous mode the 82590 receives all frames. When the address match is successful, the 82590 transfers the frame to memory using the DMA controller. Before enabling the receiver, it is the CPU's responsibility to make a memory buffer area available to the receiver and to properly program the starting address of the DMA controller. The received frame is transferred to the memory buffer in the format shown in Figure 11. This method of reception is called Single Buffer reception; the entire frame is contained in one continuous buffer. Upon completion of reception, the status of the reception is appended at the end of the received frame in the memory buffer, and the total number of bytes transferred to the memory buffer is loaded into the internal status registers 1 and 2. An interrupt is then generated to inform the CPU of the frame reception.

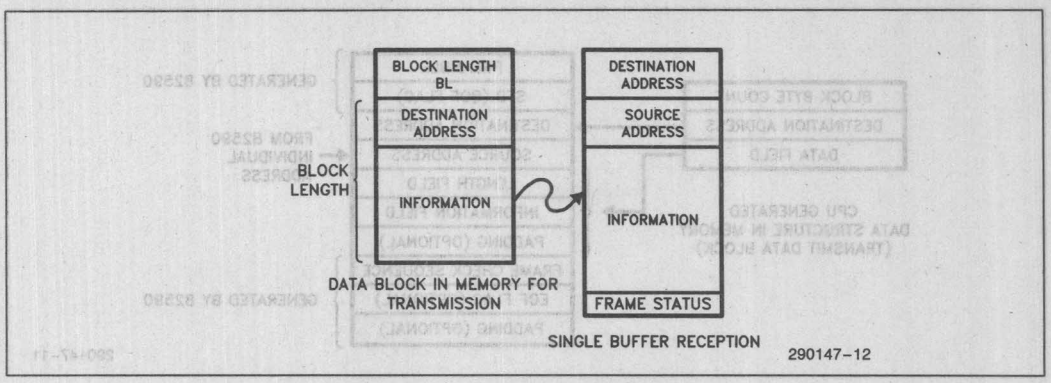


Figure 11. Single Buffer Reception

If the frame size is unknown, memory usage can be optimized by using Multiple Buffer reception. In this mode of operation, the CPU and DMA Controller can dynamically allocate memory space as it receives frames. This method requires both DMA channels to receive the frame alternately. As frame reception begins, the 82590 interrupts the CPU and automatically requests assignment of the next available buffer. The CPU does this and loads the second DMA channel with the next buffers information so the 82590 can immediately switch to the other channel when the current buffer is full. When the 82590 switches from the first to the second buffer it again interrupts the CPU and requests another buffer to be allocated on the previous channel. This process continues until the entire frame is received. The received frame is spread over multiple memory buffers. The link between the buffers is easily maintained by the CPU, using a buffer chain descriptor structure in memory as shown in Figure 12. This dynamic allocation of memory buffers results in efficient use of available storage when handling frames of widely differing sizes.

If the 82590 is programmed to generate the $\overline{EOP}$ signal to the 8237 or 82380 DMA controller, or if it is used with a DMA controller which implements the Tightly Coupled Interface, buffer reclamation and more advanced data structures for the buffer area can significantly improve system performance.

$\overline{EOP}$ SIGNAL TO THE DMA CONTROLLER

The 82590 can be programmed to assert the $\overline{EOP}$ signal to the 8237 or 82380 DMA controller when one or more of the following occurs:

- A collision during transmission
- An error (CRC or alignment) during reception
- A good frame reception

If the 8237 or 82380 is programmed for Auto-initialize mode and if the 82590 is programmed to assert the $\overline{EOP}$ signal on a collision during transmission, the retransmission following a collision is done automatically by the 8237 and the 82590. The 8237 will reinitialize itself automatically and the 82590 will retransmit the same frame from the same memory area without CPU intervention. When the 82590 is programmed for this mode it does not interrupt the CPU upon a collision, and the CPU does not need to issue a Retransmit command to the 82590. The CPU is interrupted only after a successful transmission or retransmission, or after a transmission failure, such as DMA underrun.

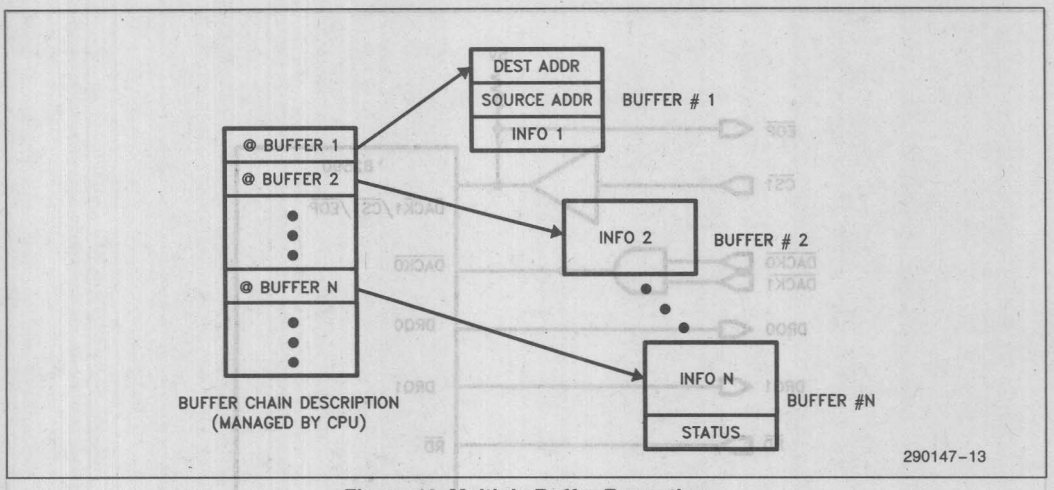


Figure 12. Multiple Buffer Reception

If the 82590 is programmed to assert the $\overline{EOP}$ signal when an error occurs during reception, the 8237 or the 82380 in Auto-initialize mode will be able to reclaim the memory area which would otherwise be wasted for the errored frame reception. If the 82590 is programmed to assert $\overline{EOP}$ at the end of a frame reception, automatic buffer switching can be accomplished by alternating the DMA channels with the 8237 or the 82380. When the 82380 is used, the buffer switching can be done with only one DMA channel.

The $\overline{EOP}$ signal must be derived from the $\overline{DACK1}$/$\overline{CS1}$/$\overline{EOP}$ pin using external logic (see Figure 13).

82590/82560 TIGHTLY COUPLED INTERFACE

The 82590 has a mode of operation called "Tightly Coupled Interface." In this mode the 82590 provides a tightly coupled interface to a DMA controller in order to execute some of the time-critical processes of the CSMA/CD protocol without any CPU intervention. By using the 82590's companion chip, the 82560, or by implementing the Tightly Coupled Interface in a DMA controller, operations such as automatic retransmission, continuous back-to-back frame reception, and transmit and/or receive buffer chaining can be accomplished.

The 82590 provides the status of the current active transmission or reception to the DMA controller by using the DRQ and $\overline{EOP}$ signals at the end of every DMA cycle. The status is encoded according to Table 3. As long as the 82590 generates DRQ High and $\overline{EOP}$ Floating at the rising edge of $\overline{RD}$ or $\overline{WR}$, the DMA controller repeats DMA transfers. If the transmission is completed without collisions or if the reception is good (no collision, no CRC, or no Alignment error), then DRQ and $\overline{EOP}$ both become Low at the end of a DMA transfer which follows the last DMA data transfer. If the transmission encountered a collision or if the reception had an error, DRQ becomes High and $\overline{EOP}$ becomes Low. The DMA controller must decode these signals appropriately and must reinitialize the DMA channel so it can retransmit the same frame or reclaim the otherwise wasted buffer. It is the DMA controller's responsibility to reprogram itself for the next appropriate operation.

The 82560 fully implements the Tightly Coupled Interface and provides very high-performance DMA services for the 82590 with minimal CPU involvement.

NETWORK MANAGEMENT AND DIAGNOSTICS

The 82590 provides a large set of diagnostic and network management functions including: internal and external loopback, monitor mode, optional capture of all frames regardless of destination address (Promiscuous mode), and time domain reflectometry for locating fault points in the network cable. The 82590 Dump command ensures software reliability by dumping the contents of the 82590 internal registers into the system memory.

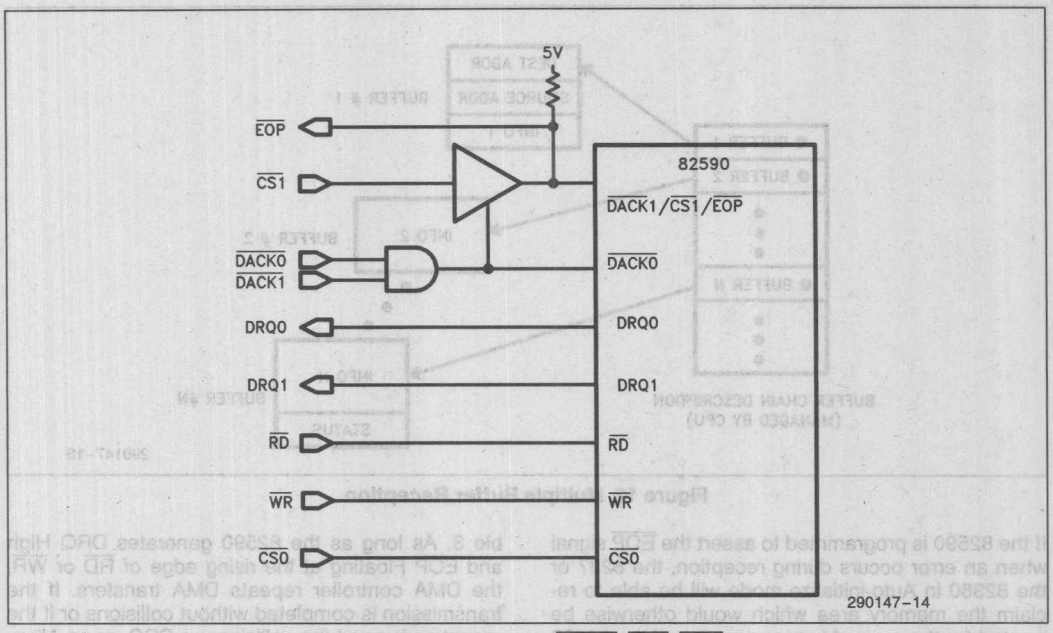


Figure 13. Demultiplexing DACK/CS1/EOP Pin

Table 3. Transmit/Receive Status Encoding on DRQ and EOP

DRQ	EOP	Status Information
0	Hi-Z	Idle
1	Hi-Z	DMA Transfer
0	0	Transmission or Reception Terminated OK
1	0	Transmission or Reception Aborted

OTHER ENHANCEMENTS

Compared to the 82588 the 82590 has a number of functional and performance enhancements. This section lists some of these enhancements which are not covered in other sections.

① **Multi-IA**—The 82590 implements multiple-individual address (Multi-IA) filtering. It can receive more than one IA frame in this mode.

② **Power Down Modes**—Two power down modes, Local Power Down and Remote Power Down, are available. When the 82590 is in Remote Power Down mode, it can be powered up remotely by sending a special frame to it.

③ **Automatic Padding and IEEE 802.3 Length Field**—If a frame to be transmitted is shorter than the configured Slot Time, the 82590 automatically appends pad bytes up to the shortest frame greater than the Slot Time. If the data field of a received frame is longer than the byte count indicated in the Length field, the extra bytes are stripped automatically according to the Length field. Erroneous conditions are detected and reported by the 82590. An example of such conditions is reception of a frame which is shorter than the byte count indicated in the Length field.

④ **Automatic Retransmission on Collision During Preamble**—The 82590 can be programmed to retransmit automatically if it detects a collision during transmission of the preamble.

⑤ **On-Chip Jabber Inhibit Function**—The 82590 can be programmed to provide an on-chip jabber inhibit function.

⑥ **CRC Transfer to Memory**—The 82590 can be programmed to transfer the CRC field of a received frame into memory.

⑦ **Loopback Signal to the 82C501**—The 82590 can be programmed to provide an active High loopback signal to the 82C501 (see Figure 14).

⑧ **StarLAN**—The 82590 can be configured to recognize the IEEE 802.3 1BASE5 Collision Presence Signal (CPS). In this mode it also delays deactivation of the $\overline{RTS}$ signal at the end of a frame transmission in order to insert an end-of-frame marker according to the standard.

APPLICATIONS

The 82590 can be used in a variety of applications. When it is used in High-Integration Mode, it implements most of the Data Link and Physical Layer functions required by the IEEE 802.3 1BASE5 (StarLAN) and the IBM PC Network—Baseband and Broadband. When it is used in High-Speed Mode, it can work with the 82C501 and a standard transceiver for IEEE 802.3 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) implementations.

Figure 15 shows a block diagram of an 82590/82560 High Integration adapter board. The 82560 provides the following functions: DMA for the 82590 with Tightly Coupled Interface and dual-port memory control for the static RAM. The 82590 is configured to High-Integration mode to minimize the serial interface logic.

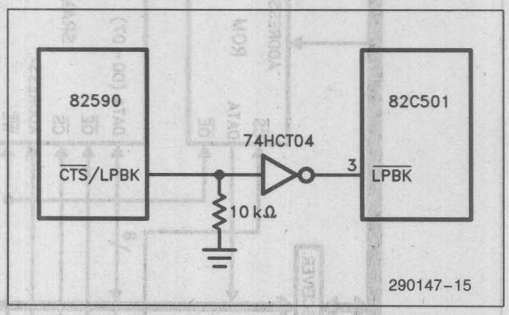


Figure 14. Loopback Output to the 82C501

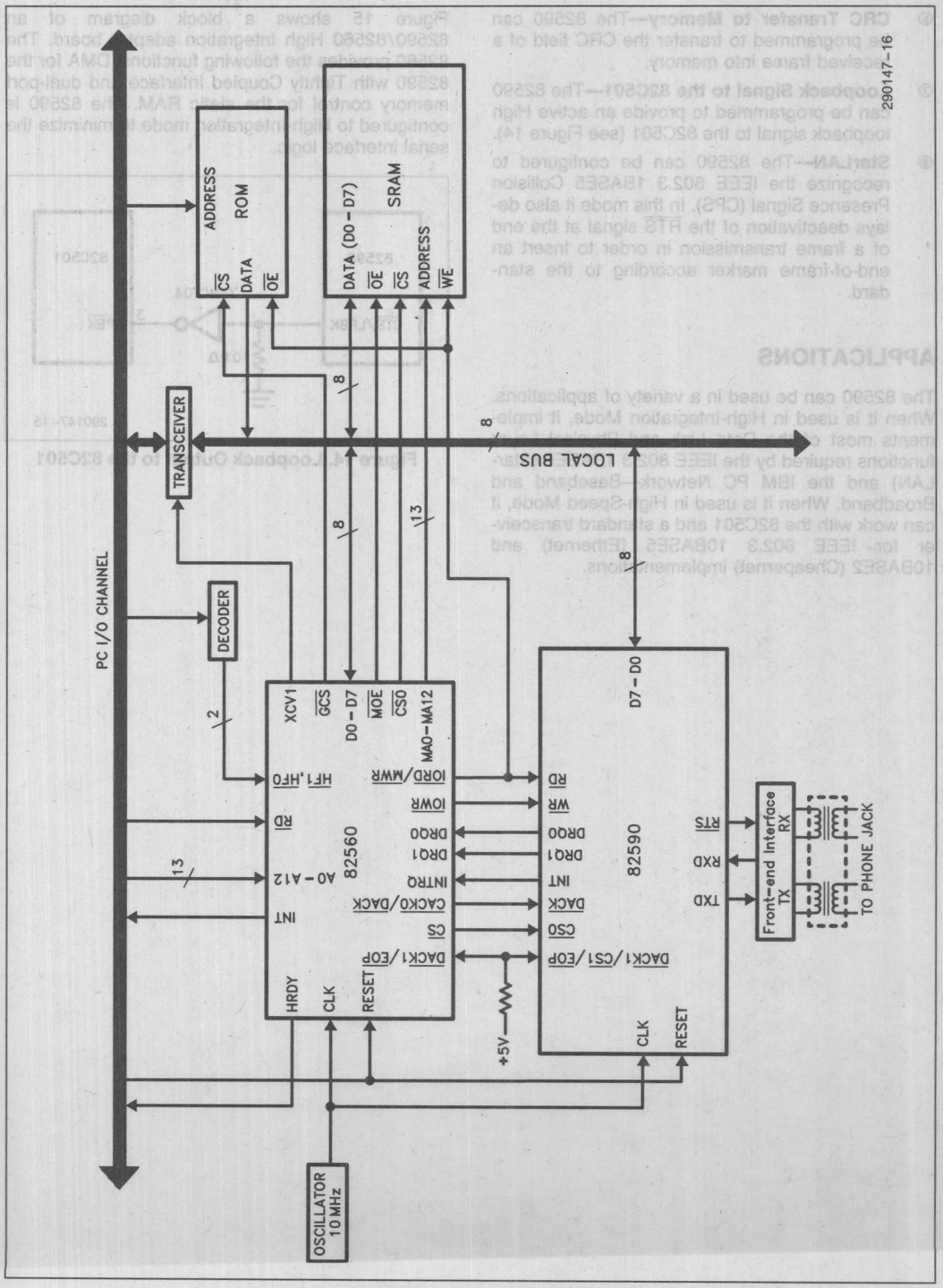


Figure 15. 82590/82560 High-Integration Adapter

ABSOLUTE MAXIMUM RATINGS*

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

Case Temperature (T_C) Under Bias
- Plastic ..... 0°C to +85°C
- PLCC ..... 0°C to +85°C

Storage Temperature ..... −65°C to +150°C

Voltage on any Pin with Respect to Ground ..... −1V to +7V

Power Dissipation ..... 550 mW

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

1

D.C. CHARACTERISTICS

(T_C [Plastic] = 0°C to +85°C, T_C [PLCC] = 0°C to +85°C, V_{CC} = +5V ±10%)

TxC and RxC have MOS levels (see V_{MIL}, V_{MIH}). All other signals have TTL levels (see V_{IL}, V_{IH}, V_{OL}, V_{OH}).

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (TTL)	−0.5	0.8	V	
V_{IH}	Input High Voltage (TTL)	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage (TTL)		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage (TTL)	2.4		V	I_{OH} = − 400 μA
V_{MIL}	Input Low Voltage (MOS)	−0.5	0.6	V	
V_{MIH}	Input High Voltage (MOS)	3.9	V_{CC} + 0.5	V	
I_{LI}	Input Leakage Current		±10	μA	$0V \le V_{IN} \le V_{CC}$
I_{LIO}	I/O Leakage Current		±10	μA	$0.45V \le V_{OUT} \le V_{CC} - 0.45V$
C_{IN}	Capacitance of Input Buffer		10	pF	Frequency = 1 MHz
C_{IO}	Capacitance of I/O Buffer		20	pF	Frequency = 1 MHz
I_{CC}	Power Supply Current		80	mA	(Note 1)
			5	mA	(Note 2)
			1	mA	(Note 3)
					(Note 4)

NOTES:
1. System side (CLK) at 16 MHz, serial side ($\overline{TxC}$) at 20 MHz in High-Speed Mode.
2. Remote power-up mode.
3. Power-down mode—all inputs connected to V_{CC} level.
4. General formulas for current are: (a) f(CLK) × 2.9 + f($\overline{TxC}$) × 1.8 for High Speed Mode. (b) f(CLK) × 2.9 + f($\overline{TFC}$) × 0.2 + f($\overline{TFC}$) × 1.8/SR, where SR is the sampling rate in High-Integration Mode, and f = frequency in MHz.

A.C. CHARACTERISTICS (C_L on all outputs is 20 pF–125 pF unless otherwise specified.)

Symbol	Parameter	Min	Max	Units	Test Conditions
SYSTEM CLOCK INPUT PARAMETERS					
t_1	CLK Cycle Time	62.5		ns	
t_2	CLK Low Time	27		ns	(Note 5)
t_3	CLK High Time	27		ns	(Note 5)
t_4	CLK Rise Time		5	ns	(Note 1)
t_5	CLK Fall Time		5	ns	(Note 2)
SYSTEM CLOCK OUTPUT PARAMETERS					
t_{98}	CLK Cycle Time	120		ns	(Notes 4, 7)
t_{99}	CLK Low Time	50		ns	(Note 4)
t_{100}	CLK High Time	50		ns	(Note 4)
t_{101}	CLK Rise Time		15	ns	(Notes 1, 4)
t_{102}	CLK Fall Time		15	ns	(Notes 2, 4)
RESET PARAMETERS					
t_6	Reset Active to Clock Low	20		ns	(Note 3)
t_8	Reset Pulse Width	$4t_1$		ns	
t_9	Control Inactive after Reset		2t1	ns	
INTERRUPT PARAMETERS					
t_{10}	CLK High to Interrupt Active		55	ns	(Note 4)
t_{11}	$\overline{WR}$ Inactive to Interrupt		55	ns	(Note 4)
t_{103}	Int Low to Int High Gap	$2t_1$		ns	(Note 4)
WRITE PARAMETERS					
t_{12}	$\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$ Setup to $\overline{WR}$ Low	0		ns	
t_{13}	$\overline{WR}$ Pulse Width	55		ns	
t_{14}	$\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$ Hold after $\overline{WR}$ High	0		ns	
t_{15}	Data Setup to $\overline{WR}$ High	30		ns	
t_{16}	Data Hold after $\overline{WR}$ High	3		ns	
t_{94}	Write Cycle Time	$2t_1$		ns	
t_{96}	$\overline{WR}$ Inactive Time	55		ns	

NOTE:
To achieve socket compatibilty with the Intel 82588, the rise and fall time specifications of the Intel 82588 can be applied for the 82590 clock inputs (CLK, $\overline{TXC}$, $\overline{RXC}$, $\overline{TFC}$). This is valid only for standard 82588 operating frequencies. Refer to the Intel 82588 data sheet for these specifications.

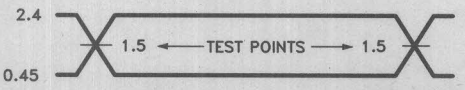
A.C. CHARACTERISTICS (C_L on all outputs is 20 pF–125 pF unless otherwise specified.) (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
READ PARAMETERS					
t_{17}	$\overline{CS0}$, $\overline{CS1}$, $\overline{DACK0}$, or $\overline{DACK1}$ Setup to $\overline{RD}$ Low	0		ns	
t_{18}	$\overline{RD}$ Pulse Width	55		ns	
t_{19}	$\overline{CS0}$, $\overline{CS1}$ $\overline{DACK0}$, or $\overline{DACK1}$ Hold after $\overline{RD}$ High	0		ns	
t_{20}	$\overline{RD}$ Low to Data Valid		45	ns	
t_{21}	Data Float after $\overline{RD}$ High	5	40	ns	
t_{95}	Read Cycle Time	$2t_1$		ns	
t_{97}	$\overline{RD}$ Inactive Time	55		ns	
DMA PARAMETERS					
t_{22}	CLK Low to DRQ0 or DRQ1 Active		55	ns	(Note 4)
t_{23}	$\overline{WR}$ or $\overline{RD}$ Low to DRQ0 or DRQ1 Inactive		45	ns	(Note 4)
t_{104}	$\overline{WR}$ or $\overline{RD}$ High to DRQ0 or DRQ1 Inactive—82560 Retransmit or Receive Buffer Reclaim	2.5	65	ns	(Note 4)
t_{105}	$\overline{WR}$ or $\overline{RD}$ Low to $\overline{EOP}$ Active		45	ns	(Note 6)
t_{106}	$\overline{EOP}$ Float after $\overline{DACK0}$ or $\overline{DACK1}$ Going Inactive		40	ns	(Note 6)

NOTES:
1. 0.8V to 2.0V.
2. 2.0V to 0.8V.
3. To guarantee recognition on the next clock.
4. C_L = 50 pF.
5. Measured at 1.5V.
6. Open drain I/O pin.
7. None of the A.C. Parameters are related to the CLK output pin.

A.C. TESTING INPUT/OUTPUT WAVEFORMS

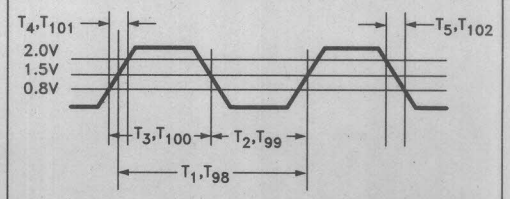
TTL Input/Output Voltage Levels for Timing Measurements



290147-17

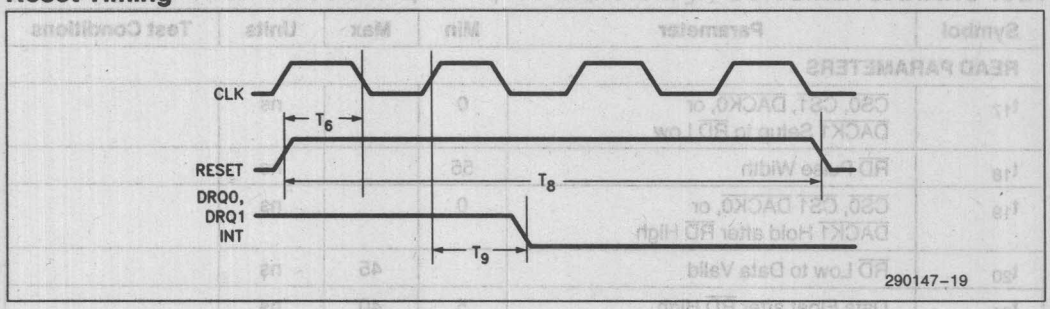
A.C. Testing inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 1.5V for both a Logic "1" and "0". Rise and fall times are measured between 0.8V and 2.0V.

System Clock I/O Voltage Levels for Timing Measurements

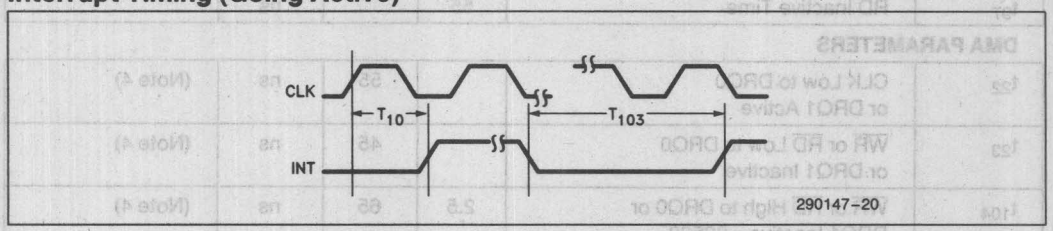


290147-18

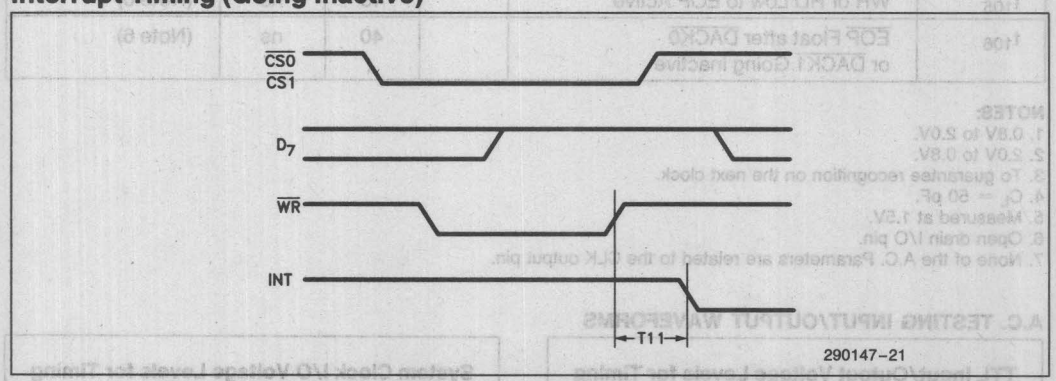
Reset Timing



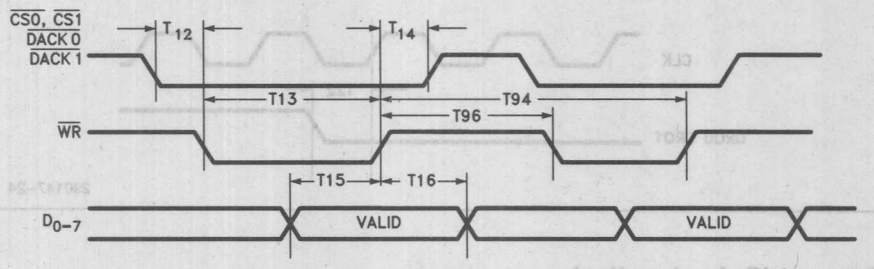
Interrupt Timing (Going Active)



Interrupt Timing (Going Inactive)



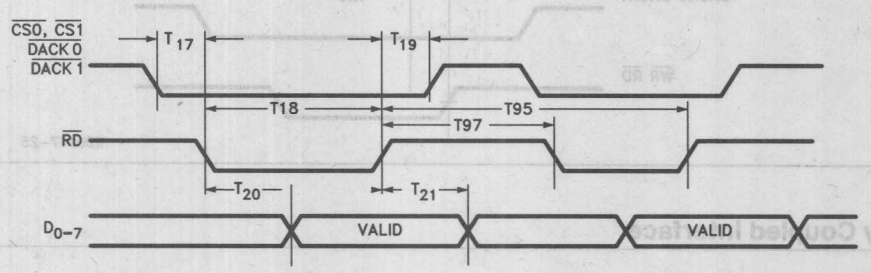
WRITE TIMING



290147-22

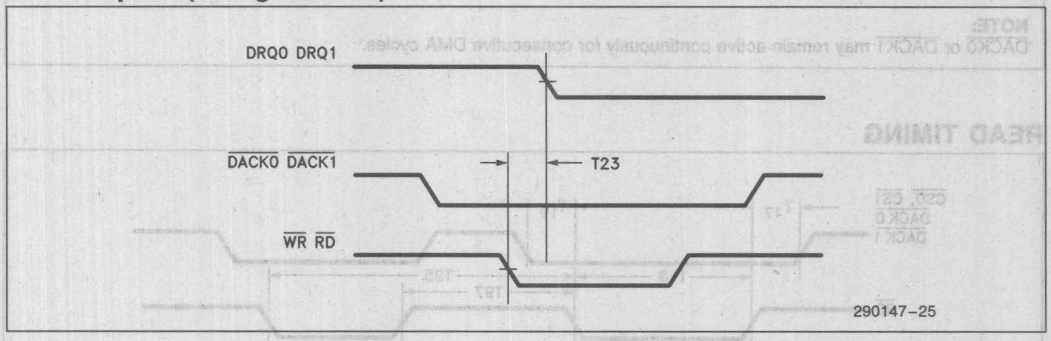
NOTE:
DACK0 or DACK1 may remain active continuously for consecutive DMA cycles.

READ TIMING

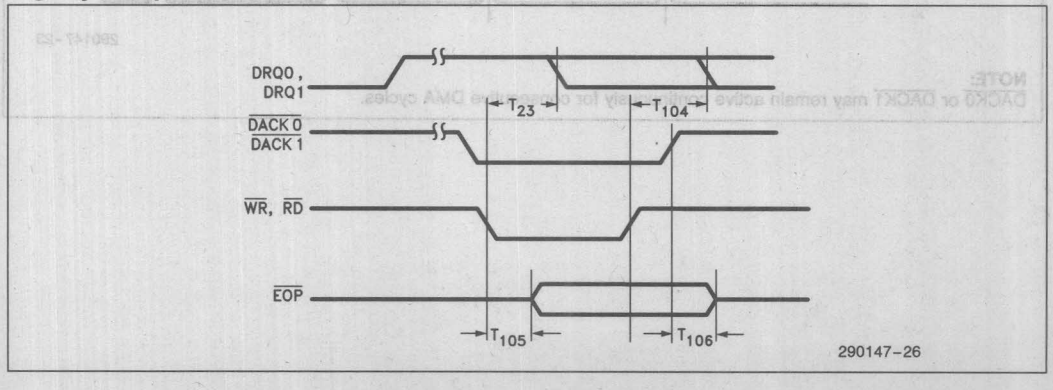


290147-23

NOTE:
DACK0 or DACK1 may remain active continuously for consecutive DMA cycles.



Tightly Coupled Interface



SERIAL INTERFACE A.C. TIMING CHARACTERISTICS

$\overline{TFC}$ is the Crystal or Serial Clock Input at X1.

	X8 Sampling	X10 Sampling	X16 Sampling	X18 Sampling
High Integration Mode For $\overline{TFC}$ Frequency = 1 MHz to 32 MHz (High)				
$\overline{TCLK}$ Frequency	0.125 MHz–4 MHz	100 kHz–3.2MHz	62.5 kHz–2 MHz	55.6 kHz–1.78 MHz
$t_{29} = \overline{TCLK}$ Cycle Time	$8 \times t_{24}$	$10 \times t_{24}$	$16 \times t_{24}$	$18 \times t_{24}$
$t_{30} = \overline{TCLK}$ High Time	t_{24} (Typically)	t_{24} (Typically)	t_{24} (Typically)	t_{24} (Typically)
$t_{31} = \overline{TCLK}$ Low Time	$7 \times t_{24}$ (Typically)	$9 \times t_{24}$ (Typically)	$15 \times t_{24}$ (Typically)	$17 \times t_{24}$ (Typically)
For $\overline{TFC}$ Frequency = 0 MHz to 1 MHz (Low)				
$\overline{TCLK}$ Frequency	0–0.125 MHz	0–100 kHz	0–62.5 kHz	0–55.6 kHz
$t_{29} = \overline{TCLK}$ Cycle Time	$8 \times t_{24}$	$10 \times t_{24}$	$16 \times t_{24}$	$18 \times t_{24}$
$t_{30} = \overline{TCLK}$ High Time	t_{25} (Typically)	t_{25} (Typically)	t_{25} (Typically)	t_{25} (Typically)
$t_{31} = \overline{TCLK}$ Low Time	$7 \times t_{24} + t_{26}$ (Typically)	$9 \times t_{24} + t_{26}$ (Typically)	$15 \times t_{24} + t_{26}$ (Typically)	$17 \times t_{24} + t_{26}$ (Typically)

NOTES:
X10 and X18 are available only for Manchester or Differential Manchester encoding/decoding
t_{24} = Serial Clock Cycle Time
t_{25} = Serial Clock High Time
t_{26} = Serial Clock Low Time

HIGH SPEED MODE

- Applies for $\overline{TxC}$, $\overline{RxC}$
- f_{max} = 20 MHz ±100 ppm
- For Manchester, symmetry is required: t_{64}, $t_{64} = \frac{1}{2f} \pm 5\%$

HIGH INTEGRATION MODE

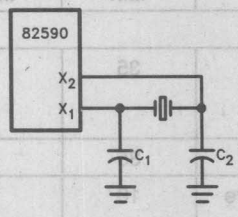
Symbol	Parameter	Min	Max	Units	Test Conditions
EXTERNAL (FAST) CLOCK PARAMETERS					
t_{24}	Fast Clock ($\overline{TFC}$) Cycle Time	31.25		ns	(Notes 1, 16)
t_{25}	$\overline{TFC}$ High Time	(Note 13)		ns	(Notes 1, 7)
t_{26}	$\overline{TFC}$ Low Time	12		ns	(Notes 1, 17)
t_{27}	$\overline{TFC}$ Rise Time		3	ns	(Note 1)
t_{28}	$\overline{TFC}$ Fall Time		3	ns	(Note 1)
TRANSMIT CLOCK PARAMETERS					
t_{29}	Transmit Clock ($\overline{TCLK}$) Cycle Time	(Note 13)		ns	(Note 2)
t_{30}	$\overline{TCLK}$ High Time	(Note 7)		ns	(Note 2)
t_{31}	$\overline{TCLK}$ Low Time	(Note 8)		ns	(Note 2)
t_{32}	$\overline{TCLK}$ Rise Time		10	ns	(Note 2)
t_{33}	$\overline{TCLK}$ Fall Time		10	ns	(Note 2)
TRANSMIT DATA PARAMETERS (MANCHESTER, DIFFERENTIAL MANCHESTER)					
t_{34}	TxD Transition-Transition	(Note 14)		ns	
t_{35}	$\overline{TCLK}$ Low to TxD Mid Bit Cell Transition		(Note 10)	ns	(Note 2)
t_{36}	$\overline{TCLK}$ Low to TxD Bit Cell Boundary Transition		(Note 9)	ns	(Note 2)
t_{37}	TxD Rise Time		10	ns	(Note 2)
t_{38}	TxD Fall Time		10	ns	(Note 2)
TRANSMIT DATA PARAMETERS (NRZI)					
t_{39}	TxD Transition-Transition	(Note 15)		ns	
t_{40}	$\overline{TCLK}$ Low to TxD Transition		(Note 9)	ns	(Note 2)
t_{41}	TxD Rise Time		10	ns	(Note 2)
t_{42}	TxD Fall Time		10	ns	(Note 2)
$\overline{RTS}$, $\overline{CTS}$ PARAMETERS					
t_{43}	$\overline{TCLK}$ Low to $\overline{RTS}$ Low		(Note 9)	ns	(Note 2)
t_{44}	$\overline{CTS}$ Low to $\overline{TCLK}$ Low	35		ns	
t_{45}	$\overline{TCLK}$ Low to $\overline{RTS}$ High		(Note 9)	ns	(Note 2)
t_{46}	$\overline{TCLK}$ Low to $\overline{CTS}$ Invalid $\overline{CTS}$ Hold Time	10		ns	(Notes 3, 12)
t_{47}	$\overline{CTS}$ High to $\overline{TCLK}$ Low; $\overline{CTS}$ Setup Time to Stop Transmission	35		ns	(Note 3)
IFS PARAMETERS					
t_{48}	Interframe Delay	(Note 4)		ns	

HIGH INTEGRATION MODE (Continued)

Symbol	Parameter	Min	Max	Units	Test Conditions
COLLISION DETECT PARAMETERS					
t_{49}	$\overline{CDT}$ Low to $\overline{TCLK}$ High External Collision Detect Setup Time	35		ns	(Note 12)
t_{50}	$\overline{CDT}$ High to $\overline{TCLK}$ High	35		ns	(Note 12)
t_{51}	$\overline{TCLK}$ High to $\overline{CDT}$ Inactive $\overline{CDT}$ Hold Time	10		ns	(Note 12)
t_{52}	$\overline{CDT}$ Low to Jamming Start		(Note 5)	ns	
t_{53}	Jamming Period	(Note 6)		ns	
RECEIVED DATA PARAMETERS (MANCHESTER, DIFFERENTIAL MANCHESTER)					
t_{54}	RxD Transition-Transition	115		ns	(Note 14)
t_{55}	RxD Rise Time		10	ns	
t_{56}	RxD Fall Time		10	ns	
RECEIVED DATA PARAMETERS (NRZI)					
t_{57}	RxD Transition-Transition	240		ns	(Note 11)
t_{58}	RxD Rise Time		10	ns	
t_{59}	RxD Fall Time		10	ns	
EXTERNAL LOOPBACK PARAMETERS					
t_{107}	$\overline{TCLK}$ Low to LPBK High		50	ns	(Note 2)
t_{108}	$\overline{TCLK}$ Low to LPBK Float		50	ns	(Note 2)

NOTES:
1. MOS Levels.
2. 1 TTL Load + 50 pF.
3. Abnormal End of Transmission: $\overline{CTS}$ expires before $\overline{RTS}$.
4. Programmable value: $t_{28} = N_{IFS} \times t_{29}$ (ns)
   N_{IFS}: the IFS configuration value.
   If N_{IFS} is less than 12 then it is enforced to 12.
5. Programmable Values:
   $t_{52} = N_{CDF} \times t_{29}$ + (12 to 15) $\times t_{29}$ (if collision occurs after preamble).
   N_{CDF}: The Collision Detect Filter Configuration Value.
6. $t_{53} = 32 \times t_{29}$
7. Depends on frequency range:
   High Range: $t_{24} - 10$ ns
   Low Range: $t_{25} - 10$ ns
8. $t_{31} = t_{29} - t_{30} - t_{32} - t_{33}$
9. $2 \times t_{24} + 40$ ns for 8X or 10X
   $4 \times t_{24} + 40$ ns for 16X
10. $6 \times t_{24} + 40$ ns for 8X
    $12 \times t_{24} + 40$ ns for 16X
    $7 \times t_{24} + 40$ ns for 10X
    $13 \times t_{24} + 40$ ns for 18X
11. $8 \times t_{24} - 10$ ns for 8X
    $10 \times t_{24} - 10$ ns for 10X
    $16 \times t_{24} - 10$ ns for 16X
    $18 \times t_{24} - 10$ ns for 18X
12. To Guarantee recognition on the next clock.
13. 10 ns for High Range
    30 ns for Low Range
14. $4 \times t_{24} - 10$ ns for 8X
    $5 \times t_{24} - 10$ ns for 10X
    $8 \times t_{24} - 10$ ns for 16X
    $9 \times t_{24} - 10$ ns for 18X
15. $t_{29} - 10$ ns
16. See Figure "CRYSTAL CONNECTION."
17. Maximum capacitance load on the X_2 pin when an external MOS clock is connected to X_1:
    15 pF for DC to 16 MHz
    5 pF for 16 MHz to 32 MHz

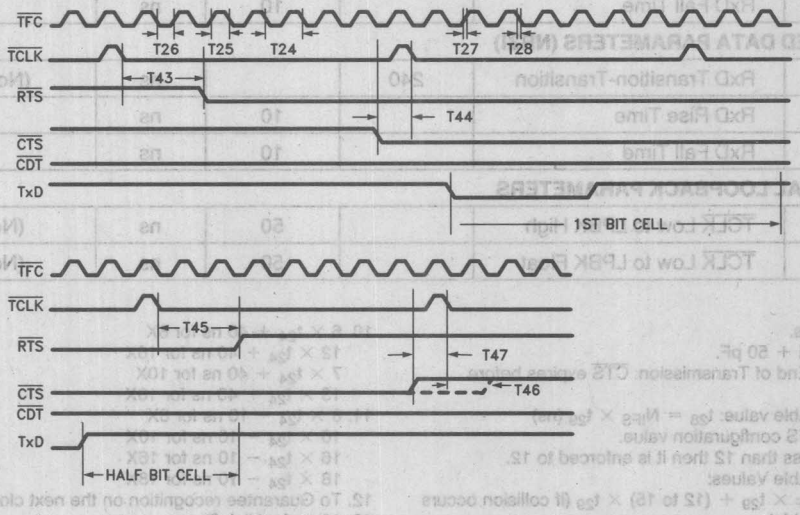
CRYSTAL CONNECTION



290147-27

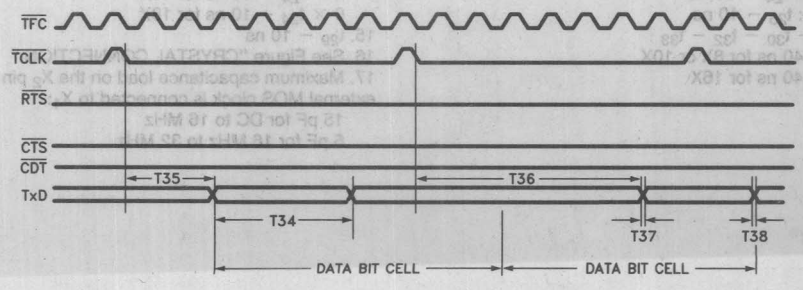
NOTES:
1. High-quality, parallel resonant, fundamental-mode crystals are recommended for maximum accuracy.
2. C1, C2, and stray capacitance of the board should be adjusted so the total capacitance load on the crystal is approx. 15 pF.
3. For IEEE 802.3 applications, the crystal must be accurate to ±35 PPM over a range of 0°C to 70°C.

TRANSMIT TIMINGS—CLOCKS $\overline{RTS}$ AND $\overline{CTS}$



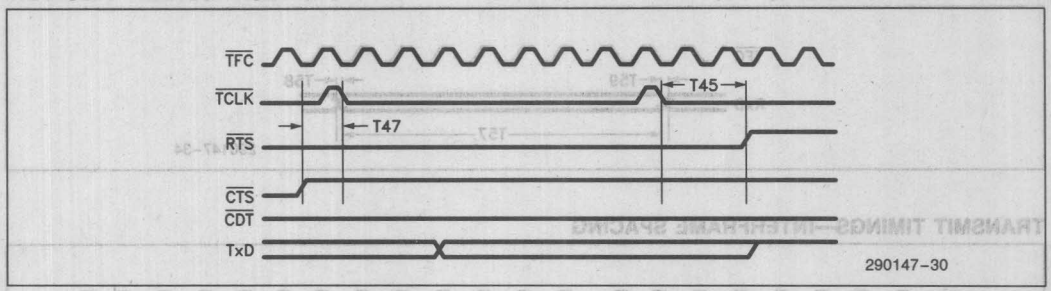
290147-28

TRANSMIT TIMINGS—MANCHESTER DATA ENCODING

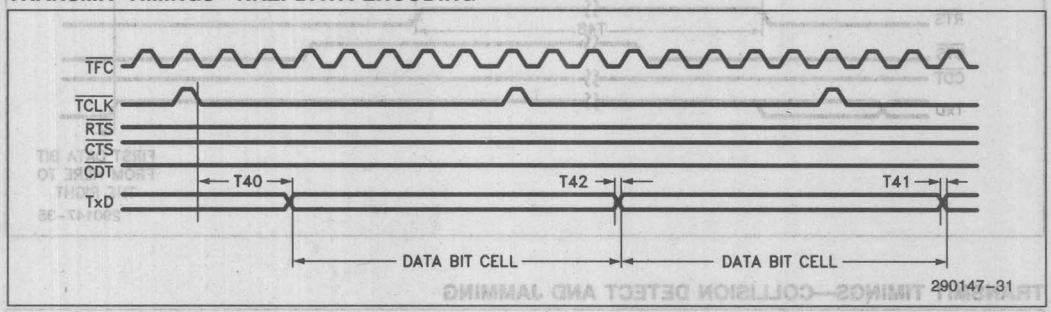


290147-29

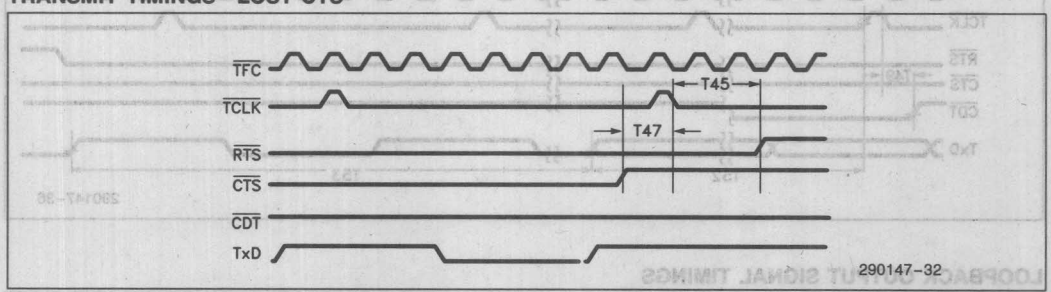
TRANSMIT TIMINGS—LOST $\overline{CTS}$



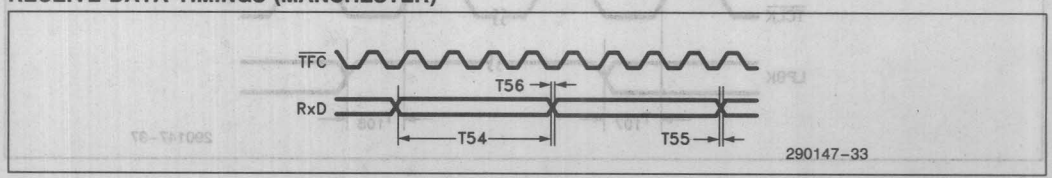
TRANSMIT TIMINGS—NRZI DATA ENCODING



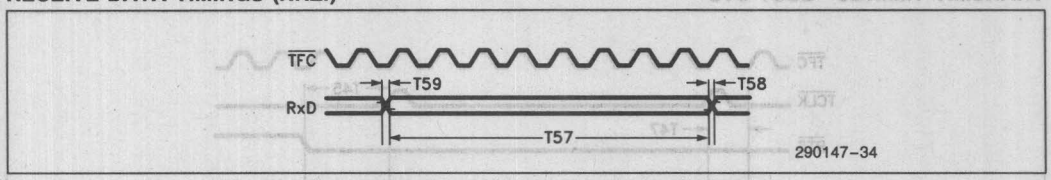
TRANSMIT TIMINGS—LOST $\overline{CTS}$



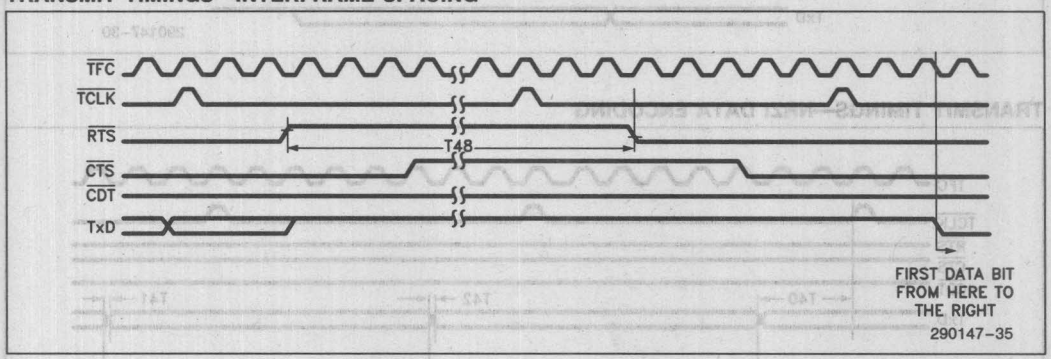
RECEIVE DATA TIMINGS (MANCHESTER)



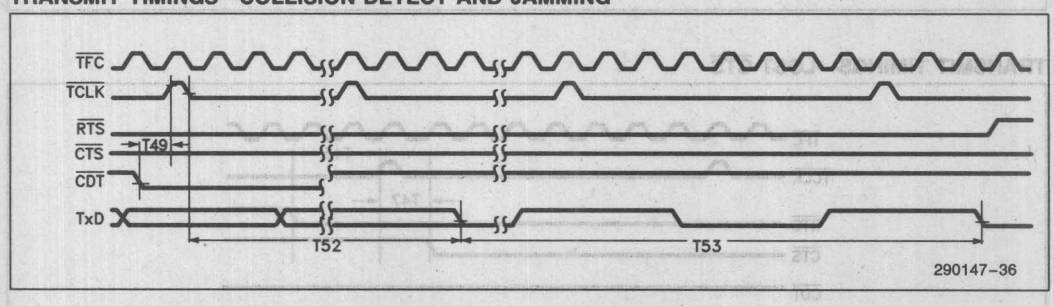
RECEIVE DATA TIMINGS (NRZI)



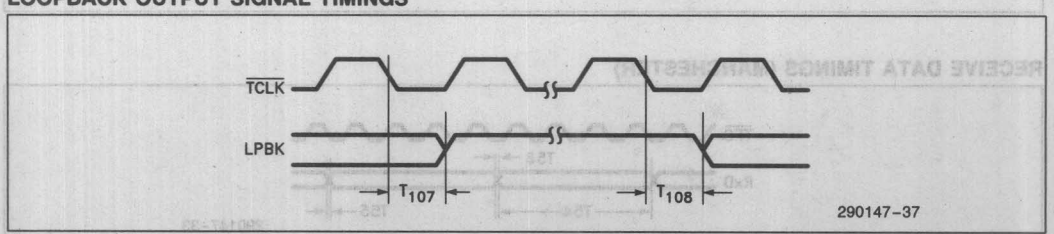
TRANSMIT TIMINGS—INTERFRAME SPACING



TRANSMIT TIMINGS—COLLISION DETECT AND JAMMING



LOOPBACK OUTPUT SIGNAL TIMINGS



HIGH SPEED MODE

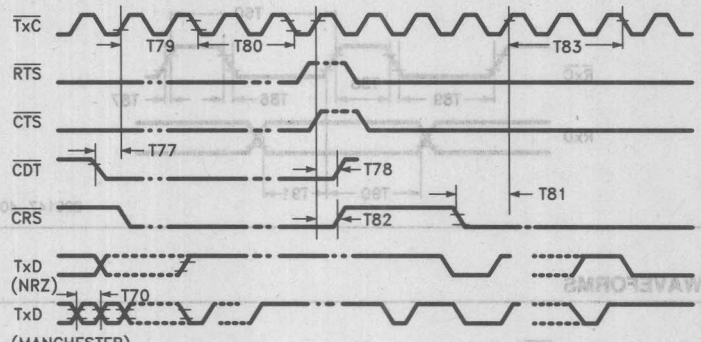
Symbol	Parameter	Min	Max	Units	Test Conditions
TRANSMIT/RECEIVE CLOCK PARAMETERS					
t_{60}	$\overline{RxC}$, $\overline{TxC}$ Cycle Time	50		ns	(Notes 1, 3)
t_{61}	$\overline{TxC}$ Rise Time		5	ns	(Note 1)
t_{62}	$\overline{TxC}$ Fall Time		5	ns	(Note 1)
t_{63}	$\overline{TxC}$ High Time	18		ns	(Notes 1, 3)
t_{64}	$\overline{TxC}$ Low Time	19		ns	(Notes 1, 3)
TRANSMIT DATA PARAMETERS					
t_{65}	TxD Rise Time		10	ns	(Note 4)
t_{66}	TxD Fall Time		10	ns	(Note 4)
t_{67}	$\overline{TxC}$ Low to TxD Valid		30	ns	(Notes 4, 5)
t_{68}	$\overline{TxC}$ Low to TxD Transition		30	ns	(Notes 2, 4)
t_{69}	$\overline{TxC}$ High to TxD Transition		30	ns	(Notes 2, 4)
t_{70}	TxD Transition-Transition	20		ns	(Notes 2, 4)
t_{71}	$\overline{TxC}$ Low to TxD High (At the Transmission End)		30	ns	(Note 4)
$\overline{RTS}$, $\overline{CTS}$ PARAMETERS					
t_{72}	$\overline{TxC}$ Low to $\overline{RTS}$ Low Time to Activate $\overline{RTS}$		30	ns	(Note 4)
t_{73}	$\overline{CTS}$ Low to $\overline{TxC}$ Low $\overline{CTS}$ Setup Time	20		ns	
t_{74}	$\overline{TxC}$ Low to $\overline{RTS}$ High		30	ns	(Note 4)
t_{75}	$\overline{TxC}$ Low to $\overline{CTS}$ Invalid. $\overline{CTS}$ Hold Time	10		ns	(Note 6)
t_{75a}	$\overline{CTS}$ High to $\overline{TxC}$ Low. $\overline{CTS}$ Setup Time to Stop Transmission	20		ns	(Note 6)
INTERFRAME SPACING PARAMETER					
t_{76}	Inter Frame Delay	(Note 8)		ns	

HIGH SPEED MODE (Continued)

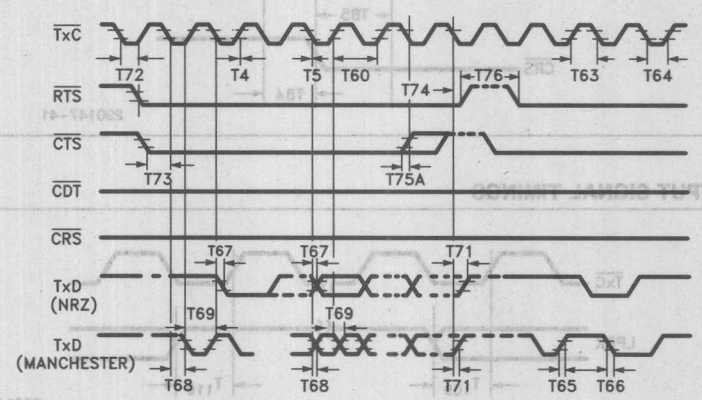
Symbol	Parameter	Min	Max	Units	Test Conditions
$\overline{CRS}$, $\overline{CDT}$ PARAMETERS					
t_{77}	$\overline{CDT}$ Low to $\overline{TxC}$ High; External Collision Detect Setup Time	20		ns	
t_{78}	$\overline{TxC}$ High to $\overline{CDT}$ Inactive; $\overline{CDT}$ Hold Time	10		ns	(Note 12)
t_{79}	$\overline{CDT}$ Low to Jam Start		(Note 9)	ns	
t_{80}	Jamming Period		(Note 10)	ns	
t_{81}	$\overline{CRS}$ Low to $\overline{TxC}$ High; Carrier Sense Setup Time	25		ns	
t_{82}	$\overline{TxC}$ High to $\overline{CRS}$ Inactive; $\overline{CRS}$ Hold Time	10		ns	(Note 12)
t_{83}	$\overline{CRS}$ High to Jamming Start (Internal Collision Detect)		(Note 11)	ns	
t_{84}	$\overline{CRS}$ High to $\overline{RxC}$ High; $\overline{CRS}$ Inactive Setup Time	30		ns	
t_{85}	$\overline{RxC}$ High to $\overline{CRS}$ High; $\overline{CRS}$ Inactive Hold Time	10		ns	
RECEIVE CLOCK PARAMETERS					
t_{86}	$\overline{RxC}$ Rise Time		5	ns	(Note 1)
t_{87}	$\overline{RxC}$ Fall Time		5	ns	(Note 1)
t_{88}	$\overline{RxC}$ High Time	18		ns	(Note 1)
t_{89}	$\overline{RxC}$ Low Time	19		ns	(Note 1)
RECEIVED DATA PARAMETERS					
t_{90}	$\overline{RxD}$ Setup Time	15		ns	(Note 5)
t_{91}	$\overline{RxD}$ Hold Time	15		ns	(Note 5)
t_{92}	$\overline{RxD}$ Rise Time		10	ns	
t_{93}	$\overline{RxD}$ Fall Time		10	ns	
EXTERNAL LOOPBACK PARAMETERS					
t_{109}	$\overline{TxC}$ Low to LPBK High		t_{60}	ns	(Note 4)
t_{110}	$\overline{TxC}$ Low to LPBK High		t_{60}	ns	(Note 4)

NOTES:
1. MOS Levels.
2. Manchester Only.
3. Manchester. Needs 50% duty cycle.
4. 1 TTL Load + 50 pF.
5. NRZ only.
6. Abnormal End of Transmission: $\overline{CTS}$ expires before $\overline{RTS}$.
7. Normal End of Transmission.
8. Programmable value:
   $t_{76} = N_{IFS} \times t_{60}$
   N_{IFS}: the IFS configuration value.
   If N_{IFS} is less than 12 then N_{IFS} is enforced to 12.
9. Programmable Value:
   $t_{79} = N_{CDF} \times t_{60}$ + (12 to 15) $\times t_{60}$ (if collision occurs after preamble).
   N_{CDF}: The collision detect filter configuration value.
10. $t_{80} = 32 \times t_{60}$
11. Programmable Value:
    $t_{83} = N_{CDF} \times t_{60}$ + (12 to 15) $\times t_{60}$
12. To guarantee recognition on the next clock.

TRANSMIT DATA WAVEFORMS

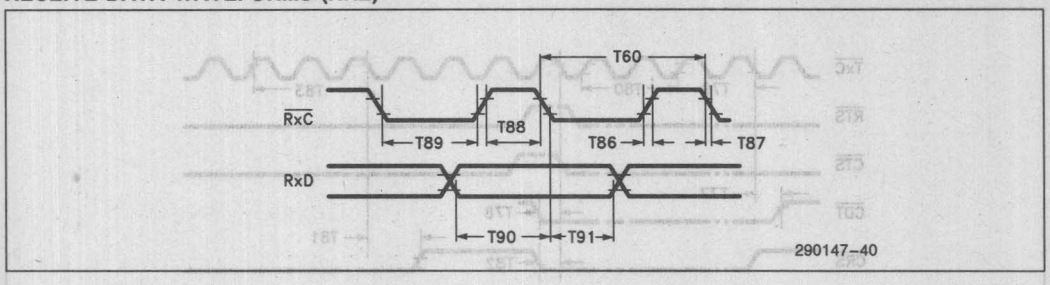


290147-38

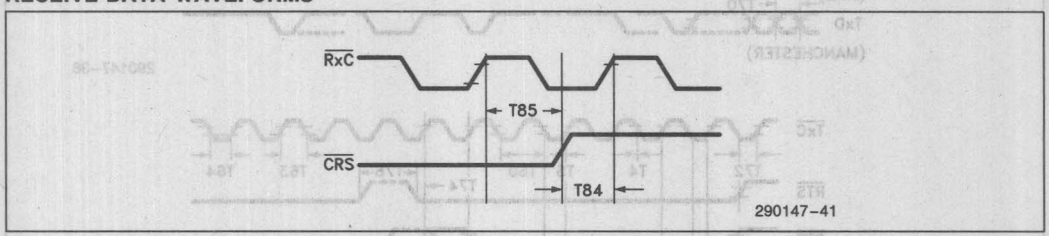


290147-39

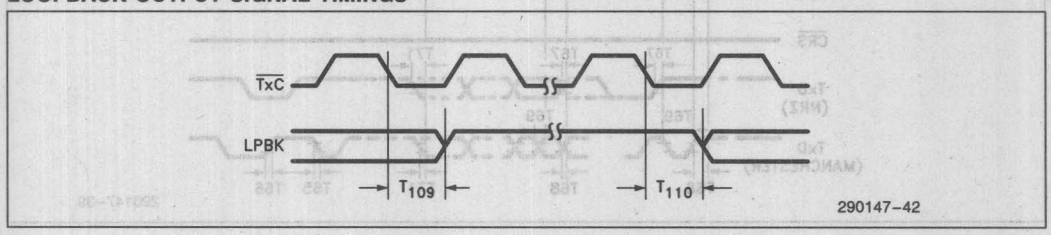
RECEIVE DATA WAVEFORMS (NRZ)



RECEIVE DATA WAVEFORMS



LOOPBACK OUTPUT SIGNAL TIMINGS



82C501AD
ETHERNET SERIAL INTERFACE

- **CHMOS Replacement for Intel 82C501, 82501, or SEEQ 8023A**
- **Conforms to IEEE 802.3 10BASE5 (Ethernet) and 10BASE2 (Cheapernet) Specifications**
- **Direct Interface to Intel LAN Controllers and the Attachment Unit Interface (Transceiver) Cable**
- **10-Mb/s Operation**
- **Manchester Encoding/Decoding and Receive Clock Recovery**
- **10-MHz Transmit Clock Generator**
- **Drives/Receives 802.3 AUI Cables**
- **Defeatable Watchdog Timer Circuit to Prevent Continuous Transmission**
- **Diagnostic Loopback for Network Node Fault Detection and Isolation**

1

The 82C501AD Ethernet Serial Interface (ESI) chip is designed to work directly with Intel LAN Controllers (82586, 82590, and 82596) in IEEE 802.3 (10BASE5 and 10BASE2), 10-Mb/s, Local Area Network applications. The major functions of the 82C501AD are to generate the 10-MHz transmit clock for the Intel LAN Controller, perform Manchester encoding/decoding of the transmitted/received frames, and provide the electrical interface to the Ethernet transceiver cable (AUI). Diagnostic loopback control enables the 82C501AD to route the signal to be transmitted from the Intel LAN Controller through its Manchester encoding and decoding circuitry and back to the Intel LAN Controller. The combined loopback capabilities of the Intel LAN Controller and 82C501AD result in highly effective fault detection and isolation through sequential testing of the communications interface. A (defeatable) on-chip watchdog timer circuit prevents the station from locking up in a continuous transmit mode. The 82C501AD is pin compatible with the 82C501 and functionally compatible with the 82501 and SEEQ 8023A.

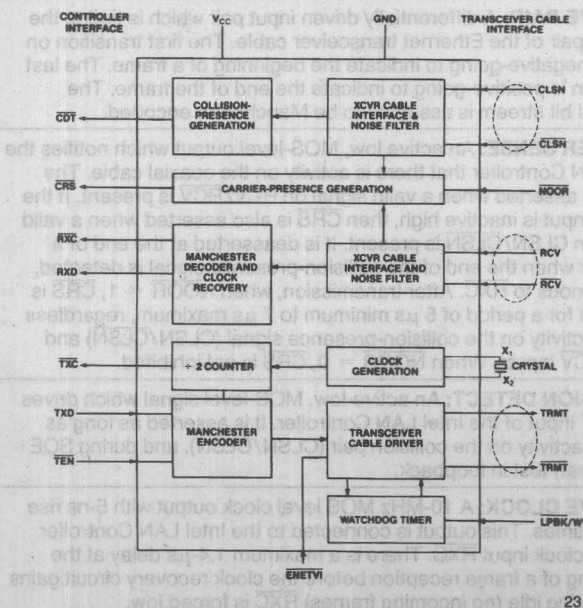


Figure 1. 82C501AD Functional Block Diagram

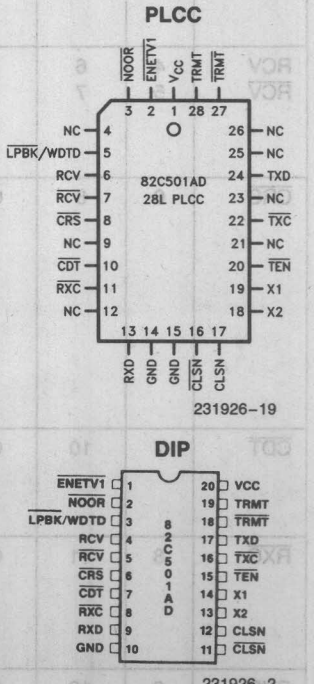


Figure 2. Pin Configurations (PLCC and DIP)

October 1990
Order Number: 231926-007

Table 1. Pin Description

Symbol	DIP Pin No.	PLCC Pin No.	Type	Name and Function
$\overline{\text{ENETV1}}$	1	2	I	**ETHERNET VERSION 1.0:** An active low, MOS-level input intended for use as a strapping option. When $\overline{\text{ENETV1}}$ is asserted, the TRMT/$\overline{\text{TRMT}}$ pair remains at high differential voltage at the end of transmission. This operation is compatible with the Ethernet Version 1.0 specification. If the $\overline{\text{ENETV1}}$ pin is left floating, an internal pull-up resistor biases the input inactive high. When $\overline{\text{ENETV1}}$ is high, the TRMT/$\overline{\text{TRMT}}$ differential voltage gradually approaches 0V at the end of transmission.
$\overline{\text{NOOR}}$	2	3	I	**CRS 'OR':** An active low, MOS-level input intended for use as a strapping option. When $\overline{\text{NOOR}}$ is low, only the presence of a valid signal on the RCV/$\overline{\text{RCV}}$ pair will force $\overline{\text{CRS}}$ active. If the $\overline{\text{NOOR}}$ pin is floating, an internal pull-up resistor biases the input inactive high. When $\overline{\text{NOOR}}$ is inactive high, either the presence of a valid signal on CLSN/$\overline{\text{CLSN}}$ or on RCV/$\overline{\text{RCV}}$ will force $\overline{\text{CRS}}$ active.
$\overline{\text{LPBK}}$/ WDTD	3	5	I	**LOOPBACK/WATCHDOG TIMER DISABLE:** An active low, TTL-level control signal that enables the loopback mode. In loopback mode serial data on the TXD input is routed through the 82C501AD internal circuits and back to the RXD output without driving the TRMT/$\overline{\text{TRMT}}$ output pair to the transceiver cable. During loopback $\overline{\text{CDT}}$ is asserted at the end of each transmission to simulate the SQE test. The $\overline{\text{LPBK}}$ signal should be driven high once V_{CC} is stabilized. **WATCHDOG TIMER DISABLE:** An input voltage of 10 to 16 V through a 1 kΩ resistor will disable the on-chip watchdog timer.
RCV $\overline{\text{RCV}}$	4 5	6 7	I I	**RECEIVE PAIR:** A differentially driven input pair which is tied to the receive pair of the Ethernet transceiver cable. The first transition on RCV is negative-going to indicate the beginning of a frame. The last transition is positive-going to indicate the end of the frame. The received bit stream is assumed to be Manchester encoded.
$\overline{\text{CRS}}$	6	8	O	**CARRIER SENSE:** An active low, MOS-level output which notifies the Intel LAN Controller that there is activity on the coaxial cable. The signal is asserted when a valid signal on RCV/$\overline{\text{RCV}}$ is present. If the $\overline{\text{NOOR}}$ input is inactive high, then $\overline{\text{CRS}}$ is also asserted when a valid signal on CLSN/$\overline{\text{CLSN}}$ is present. It is deasserted at the end of a frame or when the end of the collision-presence signal is detected, synchronous to $\overline{\text{RXC}}$. After transmission, when $\overline{\text{NOOR}}$ = 1, $\overline{\text{CRS}}$ is inhibited for a period of 5 μs minimum to 7 μs maximum, regardless of any activity on the collision-presence signal (CLSN/$\overline{\text{CLSN}}$) and RCV/$\overline{\text{RCV}}$ inputs. When $\overline{\text{NOOR}}$ = 0, $\overline{\text{CRS}}$ is not inhibited.
$\overline{\text{CDT}}$	7	10	O	**COLLISION DETECT:** An active-low, MOS-level signal which drives the $\overline{\text{CDT}}$ input of the Intel LAN Controller. It is asserted as long as there is activity on the collision pair (CLSN/$\overline{\text{CLSN}}$), and during SQE (heartbeat) test in loopback.
$\overline{\text{RXC}}$	8	11	O	**RECEIVE CLOCK:** A 10-MHz MOS level clock output with 5-ns rise and fall times. This output is connected to the Intel LAN Controller receive clock input $\overline{\text{RXC}}$. There is a maximum 1.4-μs delay at the beginning of a frame reception before the clock recovery circuit gains lock. During idle (no incoming frames) $\overline{\text{RXC}}$ is forced low.
RXD	9	13	O	**RECEIVE DATA:** A MOS-level output tied directly to the RXD input of the Intel LAN Controller and sampled by the Intel LAN Controller at the negative edge of $\overline{\text{RXC}}$. The bit stream received from the transceiver cable is Manchester decoded prior to being transferred to the controller. This output remains high during idle.

Table 1. Pin Description (Continued)

Symbol	DIP Pin No.	PLCC Pin No.	Type	Name and Function
GND	10	14 15		**GROUND**
CLSN $\overline{CLSN}$	12 11	17 16	I I	**COLLISION PAIR:** A differentially driven input pair tied to the collision-presence pair of the Ethernet transceiver cable. The collision-presence signal is a 10-MHz square wave. The first transition at CLSN is negative-going to indicate the beginning of the signal; the last transition is positive-going.
X_1 X_2	14 13	19 18	I I	**CLOCK CRYSTAL:** 20-MHz crystal inputs. When X_2 is floated, X_1 can be driven by an external MOS level input clock.
$\overline{TEN}$	15	20	I	**TRANSMIT ENABLE:** An active low, TTL level signal synchronous to $\overline{TXC}$ that enables data transmission to the transceiver cable and starts the watchdog timer. $\overline{TEN}$ can be driven by the $\overline{RTS}$ signal from the Intel LAN Controller.
$\overline{TXC}$	16	22	O	**TRANSMIT CLOCK:** A 10-MHz MOS level clock output with 5-ns rise and fall times. This clock is connected directly to the $\overline{TXC}$ input of the Intel LAN Controller.
TXD	17	24	I	**TRANSMIT DATA:** A TTL-level input signal that is directly connected to the serial data output, TXD, of the Intel LAN Controller.
TRMT $\overline{TRMT}$	19 18	28 27	O O	**TRANSMIT PAIR:** A differential output driver pair that drives the transmit pair of the transceiver cable. The output bit stream is Manchester encoded. Following the last transmission, which is always positive at TRMT, the differential voltage is slowly reduced to zero volts in a series of steps. If $\overline{ENETV1}$ is asserted this voltage stepping is disabled.
V_{CC}	20	1		**POWER:** 5V ±10%.

FUNCTIONAL DESCRIPTION

Clock Generation

A 20 MHz parallel resonant crystal is used to control the clock generation oscillator, which provides the basic 20 MHz clock source. An internal divide-by-two counter generates the 10 MHz ± 0.01% clock required by the IEEE 802.3 specification.

It is recommended that a crystal meeting the following specifications be used:

- Quartz Crystal
- 20.00 MHz ± 0.002% at 25°C
- Accuracy ±0.005% Over Full Operating Temperature, 0 to 70°C
- Parallel Resonant with 20-pF Load Fundamental Mode

Several vendors have these crystals available; either off the shelf or custom made. Two possible vendors are:

1. M-Tron Industries, Inc
   Yankton, SD 57078
2. Crystek Corporation
   100 Crystal Drive
   Ft Myers, FL 33907

The accuracy of the Crystal Oscillator frequency depends on the PC board characteristics, therefore it is advised to keep the X1 and X2 traces as short as possible. The optimum value of C1 and C2 should be determined experimentally under nominal operating conditions. The typical value of C1 and C2 is between 22 and 35 pF.

An external, 20 MHz, MOS-level clock may be applied to pin X_1 while pin X_2 is left floating.

TRANSMIT SECTION

Manchester Encoder and Transceiver Cable Driver

The 20-MHz clock is used to Manchester encode data on the TXD input line. The clock is also divided by two to produce the 10-MHz clock required by the

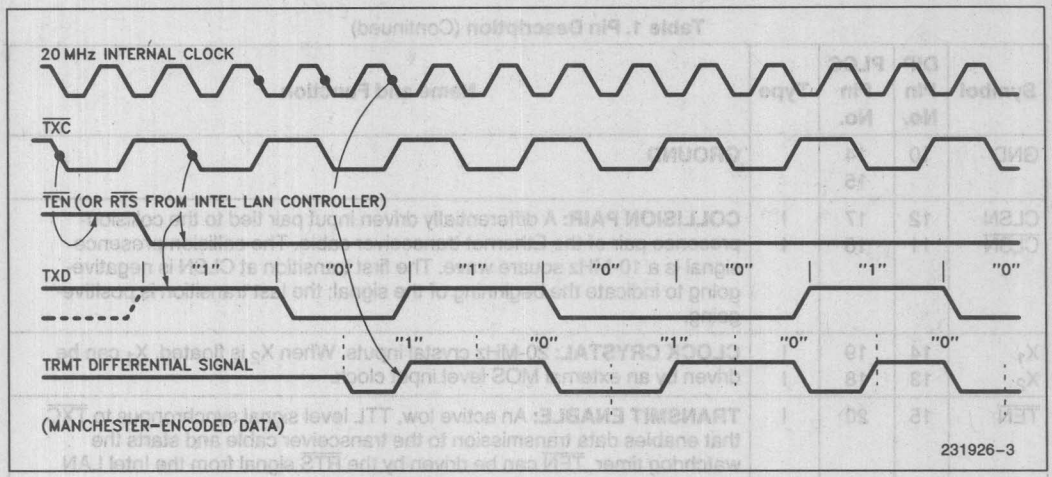


Figure 3. Start of Transmission and Manchester Encoding

Intel LAN Controller for synchronizing its $\overline{RTS}$ and TXD signals. See Figure 3. (Note that the Intel LAN Controller $\overline{RTS}$ is tied to the 82C501AD $\overline{TEN}$ input as shown in Figure 4.)

Data encoding and transmission begins with $\overline{TEN}$ going low. Since the first bit is a '1', the first transition on the transmit output TRMT is always negative. Transmission ends with the $\overline{TEN}$ going high. The last transition is always positive at TRMT and can occur at the center of the bit cell (last bit = 1) or at the boundary of the bit cell (last bit = 0). A 1.5-bit delay is introduced by the 82C501AD between its TXD input and TRMT/$\overline{TRMT}$ output as shown in Figure 3. If the signal applied to the $\overline{ENETV1}$ input is inactive high, the TRMT differential output is kept at high differential for 200 ns, after the last transmit data transition, then it is gradually reduced. The TRMT/$\overline{TRMT}$ differential voltage will become less than 40 mV within t_{18} after the last positive transition. The undershoot for return to idle is less than 100 mV differentially. This mode of operation is compatible with the IEEE 802.3 transceiver specifications.

If an active signal is present at the $\overline{ENETV1}$ input at the end of transmission, the TRMT/$\overline{TRMT}$ pair output will remain at a high differential voltage. As a result there is a positive differential voltage during the entire transmit idle time. This mode of operation is compatible with the Ethernet Version 1.0 specification.

Immediately after the end of a transmission all signals on the receive pair are inhibited for 5 μs minimum to 7 μs maximum (when $\overline{NOOR}$ = 1). This dead time is required for proper operation of the SQE (heartbeat) test.

An internal watchdog timer is started when $\overline{TEN}$ is asserted low at the beginning of the frame. The duration of the watchdog timer is 25 ms ± 15%. If the transmission terminates (by deasserting the $\overline{TEN}$) before the timer expires, the timer is reset (and ready for the next transmission). If the timer expires before the transmission ends, the frame is aborted. The frame is aborted by disabling the output driver for the TRMT/$\overline{TRMT}$ pair. RXD and $\overline{RXC}$ are not affected. The watchdog timer is reset only when the $\overline{TEN}$ is deasserted.

The cable driver is a differential circuit requiring external pulldown resistors of 240Ω ±5%. In addition, high-voltage protection to +10V maximum, and short circuit protection to ground is provided.

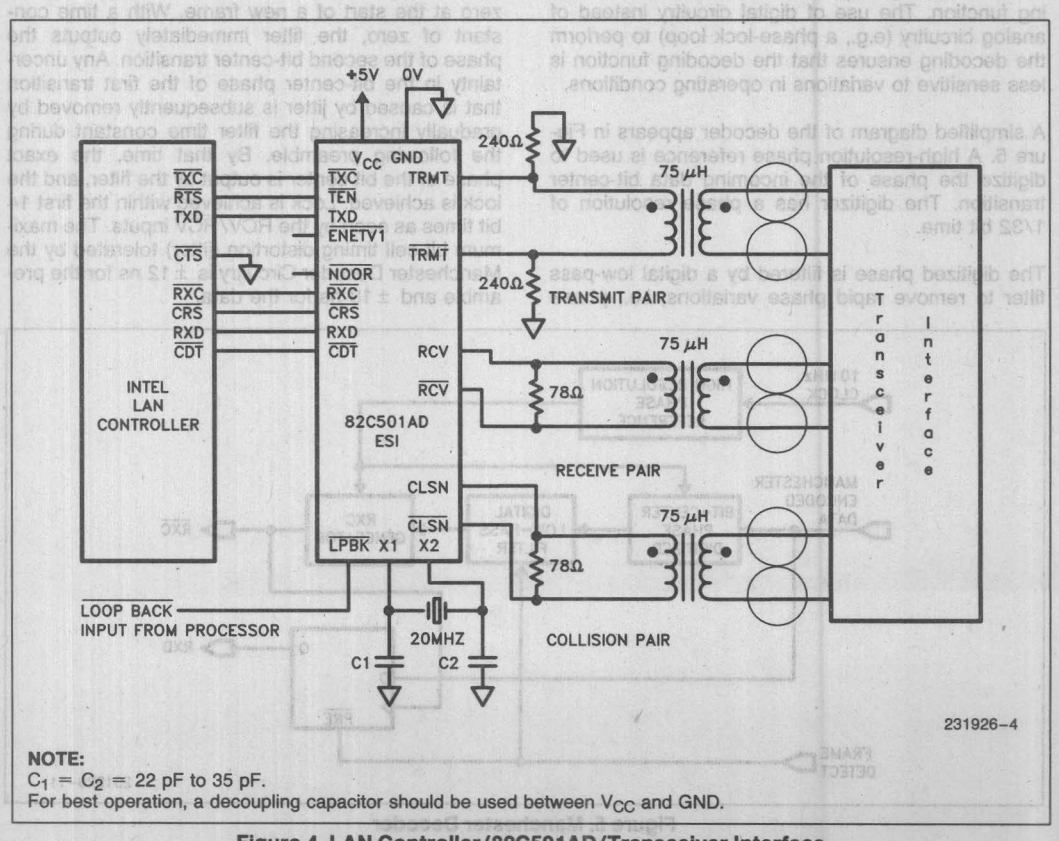
To provide additional high voltage protection if the cable is shorted, an isolation transformer can be used to isolate the TRMT and $\overline{\text{TRMT}}$ outputs from the transceiver cable. Transmit circuit inductance (including the IEEE 802.3 transceiver transformers) should be a minimum of 27 μH. We recommend that the transformer at the 82C501AD end have a minimum inductance of 75 μH.

RECEIVE SECTION

Cable Interface

The 82C501AD input circuits can be driven directly from the Ethernet transceiver cable receive pair. In this case the cable is terminated with a resistor of 78Ω ±6% for proper impedance matching. See Figure 4.

The signal received on the RCV/$\overline{\text{RCV}}$ pair from the transceiver defines both the $\overline{\text{RXC}}$ and RXD outputs to the Intel LAN Controller. The $\overline{\text{RXC}}$ and RXD signals are recovered from the encoded RCV/$\overline{\text{RCV}}$ pair signal by the Manchester decode circuitry.



NOTE:
$C_1 = C_2 = 22$ pF to 35 pF.
For best operation, a decoupling capacitor should be used between V_{CC} and GND.

Figure 4. LAN Controller/82C501AD/Transceiver Interface

The input circuits can also be driven with ECL voltage levels. In either case, the input common mode voltage must be in the range of 0-V_{CC} volts to allow for wide driver supply variation at the transceiver. To provide additional high voltage protection, if the cable is shorted, an isolation transformer can be used to isolate the RCV and $\overline{RCV}$ inputs from the cable.

Manchester Decoder and Clock Recovery

The Manchester-encoded data stream is decoded to separate the Receive Clock ($\overline{RXC}$) and the Receive Data ($\overline{RXD}$) from the stream. The 82C501AD uses an advanced digital technique to perform the decoding function. The use of digital circuitry instead of analog circuitry (e.g., a phase-lock loop) to perform the decoding ensures that the decoding function is less sensitive to variations in operating conditions.

A simplified diagram of the decoder appears in Figure 5. A high-resolution phase reference is used to digitize the phase of the incoming data bit-center transition. The digitizer has a phase resolution of 1/32 bit time.

The digitized phase is filtered by a digital low-pass filter to remove rapid phase variations; i.e., phase jitter. Slow phase variations, such as those caused by small differences between the data frequency and the clock frequency, are passed unfiltered by the low-pass filter.

The $\overline{RXC}$ generator digitally sets the phases of the two $\overline{RXC}$ transitions to respectively lead and lag the bit-center transition by 1/4 bit time. $\overline{RXC}$ is used to recover RXD by sampling the incoming data with an edge-triggered flip-flop.

The Frame_Detect signal informs the decoder that the first valid negative transition of a new frame has been detected. This signal is used to initiate the lock-on sequence of the decoder. Lock is achieved by reducing the time constant of the digital filter to zero at the start of a new frame. With a time constant of zero, the filter immediately outputs the phase of the second bit-center transition. Any uncertainty in the bit-center phase of the first transition that is caused by jitter is subsequently removed by gradually increasing the filter time constant during the following preamble. By that time, the exact phase of the bit center is output by the filter, and the lock is achieved. Lock is achieved within the first 14 bit times as seen by the RCV/$\overline{RCV}$ inputs. The maximum bit-cell timing distortion (jitter) tolerated by the Manchester Decoder Circuitry is ±12 ns for the preamble and ±18 ns for the data.

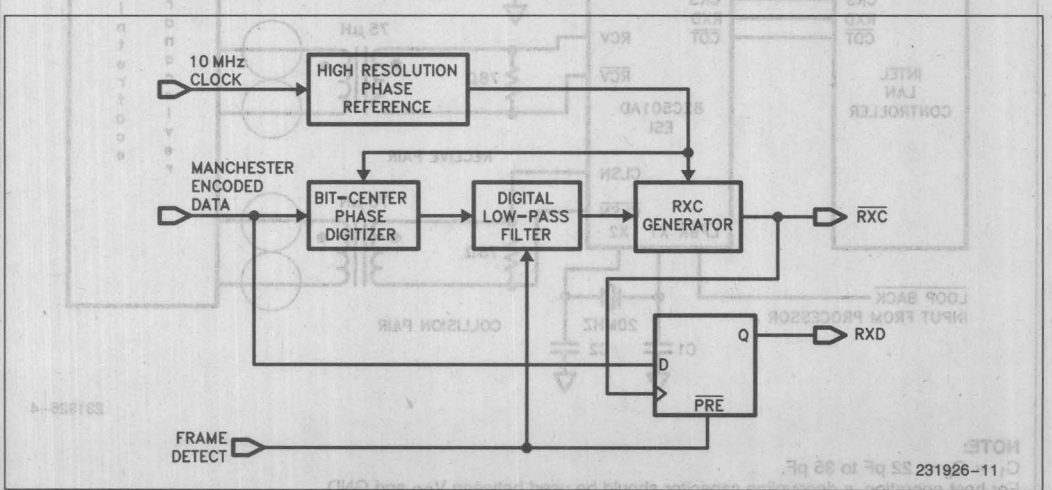


Figure 5. Manchester Decoder

COLLISION-PRESENCE SECTION

The CLSN/$\overline{\text{CLSN}}$ input signal is a 10 MHz +25%/−15% square-wave generated by the transceiver whenever two or more data frames are superimposed on the coaxial cable. The pulse width of the CLSN/$\overline{\text{CLSN}}$ signal can be no less than 35 ns and no greater than 70 ns measured at the 0-V crossing.

The common-mode voltage and external termination are identical to the RCV/$\overline{\text{RCV}}$ input. (See Figure 4.)

A valid collision presence signal will assert the 82C501AD $\overline{\text{CDT}}$ output, which can be directly tied to the $\overline{\text{CDT}}$ input of the Intel LAN Controller. During normal operation the 82C501AD logically "ORs" the collision presence signal with an internal signal, indicating valid data reception on the RCV/$\overline{\text{RCV}}$ pair, to generate $\overline{\text{CRS}}$ output. If, however, the $\overline{\text{NOOR}}$ input is asserted low, this "OR" function is removed and $\overline{\text{CRS}}$ is only asserted by the presence of valid data on the RCV/$\overline{\text{RCV}}$ pair. This mode of operation is required for repeater design.

During the time that valid collision-presence transitions are present on the CLSN/$\overline{\text{CLSN}}$ input, invalid data transitions may be present on the receive data pair due to the superposition of signals from two or more stations transmitting simultaneously. It is possible for RCV/$\overline{\text{RCV}}$ to lose transitions for a few bit times due to perfect cancellation of the signals; this may cause the 82C501AD to abort the reception.

The $\overline{\text{CRS}}$ signal is asserted low (along with $\overline{\text{CDT}}$) whenever a valid collision-presence signal is present and $\overline{\text{NOOR}}$ = 1. If this collision-presence signal arrives within 5 μs to 7 μs after the last transmission, only $\overline{\text{CDT}}$ is generated. This ensures that the LAN Controller recognizes the active $\overline{\text{CDT}}$ as a valid SQE (heartbeat) test signal.

NOISE FILTERING ON RCV AND CLSN PAIRS

Both the receive and collision pairs have the following characteristics.

- At idle, the noise filter is turned on.
- A pulse is rejected if:

a. Its peak voltage is more positive than −150 mV, with no restriction on width, or:

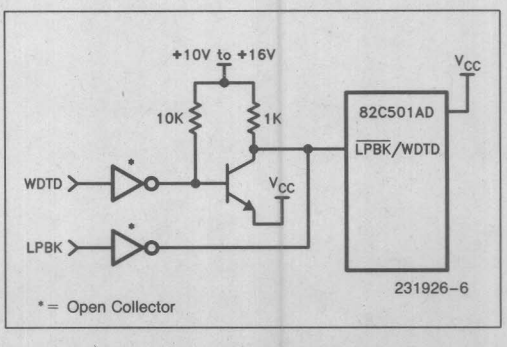
b. Its peak voltage is more positive than −600 mV and its width is less than 5 ns (measured at a reference level of −285 mV).

- The filter is turned off by the first valid negative pulse on the RCV or CLSN pair. A pulse whose peak voltage is more negative than −300 mV and whose width is greater than 30 ns (measured at −285 mV) is considered valid.
- The filter is turned on again when no positive transition is observed on the RCV or CLSN pair for 160 ns.

Internal Loopback

When asserted, $\overline{\text{LPBK}}$ causes the 82C501AD to route serial data from its TXD input through its transmit logic (retiming and Manchester encoding); returning it through the receive logic (Manchester decoding and receive clock generation) to RXD output. The internal routing prevents the data from passing through the output drivers and onto the transmit output pair TRMT/$\overline{\text{TRMT}}$. When in loopback mode all of the transmit and receive circuits, are tested except for the transceiver cable output driver and input receivers. Also, at the end of each frame transmitted in loopback mode the 82C501AD generates the SQE test (heartbeat) signal within 1 μs after the end of the frame. Thus, the collision circuits are also tested in loopback mode. During loopback, as in any normal reception, the 82C501AD receive circuitry uses 14 bit times while the Manchester Decoder locks on the data. As a result, the first 14 bits are lost and $\overline{\text{RXC}}$ is held low during that time.

The watchdog timer remains enabled in loopback mode, terminating test frames that exceed its timeout period. The watchdog timer can be inhibited by connecting $\overline{\text{LPBK}}$ to a 1 kΩ resistor connected to 10 to 16V. The loopback feature can still be used to test the integrity of the 82C501AD by using the circuit shown in Figure 6.



LPBK	WDTD	Function
1	X	LPBK mode
0	0	Normal mode
0	1	Normal mode with watchdog timer disabled

Figure 6. Watchdog Timer Disable

The 82C501AD operates as a full-duplex device, being able to transmit and receive simultaneously. By

combining the internal and external loopback modes of the Intel LAN Controller, and the internal loopback and normal modes of the 82C501AD, incremental testing of an Intel LAN Controller/82C501AD-based interface can be performed under program control for systematic fault detection and fault isolation.

Interface Example

The 82C501AD is designed to work directly with the Intel LAN Controller in IEEE 802.3 10 Mb/s, as well as other 10 Mb/s LAN applications. The control and data signals connect directly between the two devices without the need for additional external logic. The complete Intel LAN Controller/82C501AD Ethernet Transceiver interface is shown in Figure 4. The 82C501AD provides the driver and receivers needed to directly connect to the transceiver cable or requiring only terminating resistors on each input signal pair and 240Ω pull-down resistors.

It is recommended that a decoupling capacitor be used between V_{CC} and GND.

The Transmit, Receive, and Collision pairs have a maximum 10V overvoltage protection.

If additional high voltage protection is desired, a pulse transformer should be included for Ethernet applications. IEEE 802.3 10BASE5 (Ethernet) specifications require at least 16V protection for the Transmit, Receive, and Collision pairs. In 10BASE2 (Cheapernet) a pulse transformer is required to be inserted between the DTE (Intel LAN Controller/82C501AD) and the transceiver. In an Ethernet/Cheapernet design, a single transformer can be used for both connections at minimal additional cost.

The pulse transformer should have the following characteristics:

1. A minimum inductance of 75 μH.
2. 2000V isolation between primary and secondary windings.
3. 2000V isolation between primaries of separate transformers.

Since Ethernet Version 1.0 transceivers can require a positive differential on the TRMT pair during idle, check with the transceiver vendor before including the pulse transformer.

ABSOLUTE MAXIMUM RATING*

Case Temperature Under Bias .....0°C to +85°C
Storage Temperature .....−65°C to +140°C
All Output and Supply Voltages .....−0.5V to +7V
All Input Voltages .....−1.0V to +6.0V(1)
Operating Power Dissipation 0.75W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

D.C. CHARACTERISTICS $T_C = 0°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter		Min	Max
V_{IL}	Input Low Voltage	TTL MOS	−0.5V −0.5V	0.8V 0.6V
V_{IH}	Input High Voltage	TTL MOS	2.0V 3.9V	V_{CC} + 0.5V V_{CC} + 0.5V
V_{ACCEPT}	Differential Input Accept Voltage		±285 mV	
V_{REJECT}	Differential Input Reject Voltage			±150 mV
V_{CM}	Input Common Mode Voltage		0V	V_{CC}
V_{OCM}	Common Mode Output Voltage(2)		0.5V	5.0V
V_{OL}	Output Low Voltage @ I_{OL} = 4 mA			0.45V
V_{OH}	Output High Voltage (MOS) @ I_{OH} = −500 μA		3.9V	
V_{ODF}	Differential Output Voltage(2)		±0.45V	±1.2V
V_U	TRMT Pair Differential Return to Zero Undershoot(2)			−100 mV
V_{DI}	TRMT Pair Differential Idle Voltage(2)			±40 mV(5)
I_{LI}	Input Leakage Current @ V_{IN} = 0V to V_{CC}(3)			±10 μA
I_{CC}	Power Supply Current @ T_C = 85°C(4)			135 mA
I_{SP}	Short Protection Activation Current		60 mA	150 mA
I_L	Input Load Current(6)			±1 mA
C_{IN}	Input Capacitance @ fc = 1 MHz(7)			10 pF

NOTES:
1. The voltage levels for CLSN/$\overline{\text{CLSN}}$, RCV/$\overline{\text{RCV}}$ inputs are −0.75V to +10V.
2. The testing load is a 78Ω ±1% resistor in parallel with a 27 μH ±1% inductor and two 240Ω ±5% pulldown resistors.
3. Applies to TXD and $\overline{\text{TEN}}$ pins.
4. Part of the power is dissipated through the pulldown resistors connected to the TRMT/$\overline{\text{TRMT}}$ outputs.
5. Measured after t_{18} has expired.
6. Applies to RCV/$\overline{\text{RCV}}$, X_1, CLSN/$\overline{\text{CLSN}}$, $\overline{\text{LPBK}}$/WDTD, $\overline{\text{NOOR}}$, and $\overline{\text{ENETV1}}$ inputs for input voltages from 0V to V_{CC}.
7. Characterized, not tested.

A.C. MEASUREMENT CONDITIONS

1. T_C = 0°C to +85°C, V_{CC} = 5V ±10%.
2. The AC MOS, TTL and differential signals are referred to in Figures 7, 8, 9, 10 and 10A.
3. AC Loads:
   a) MOS: a 20 pF total capacitance to ground.
   b) Differential: a 10 pF total capacitance from each terminal to ground, two 240Ω ±5% pull down resistors, and a load resistor of 78Ω ±1% in parallel with a 27 μH ±1% inductor between terminals.
4. All AC Parameters become valid 100 μs after the supply voltage has stabilized.

Clock Timing(1)

Symbol	Parameter	Min	Max	Unit
t_1	X_1 Cycle Time	49.995	50.005	ns
t_2	X_1 Fall Time(2)		5	ns
t_3	X_1 Rise Time(2)		5	ns
t_4	X_1 Low Time(2)	15		ns
t_5	X_1 High Time(2)	15		ns

NOTES:
1. Refer to Figure 9.
2. Applies to external clock inputs.

TRANSMIT TIMING(1)

Symbol	Parameter	Min	Max	Unit
t_6	$\overline{TXC}$ Cycle Time(2)	99.99	100.01	ns
t_7	$\overline{TXC}$ Rise/Fall Time		5	ns
t_8	TXD Rise/Fall Time		10	ns
t_9	$\overline{TXC}$ Low Time	40		ns
t_{10}	$\overline{TXC}$ High Time	40		ns
t_{11}	Transmit Enable/Disable to $\overline{TXC}$ Low	45		ns
t_{12}	TXD Stable to $\overline{TXC}$ Low	45		ns
t_{13}	Bit Cell Center to Bit Cell Center of Transmit Pair Data(3)	99.5	100.5	ns
t_{14}	$\overline{TEN}$ Rise/Fall Time		10	ns
t_{15}	Transmit Differential Signal Rise/Fall Time		5.0	ns
t_{16}	Bit Cell Center to Bit Cell Boundary of Transmit Pair Data(3)	49.5	50.5	ns
t_{17}	$\overline{TRMT}$ held low from Last Positive Transition of the Transmit Pair at the End of Frame	200		ns
t_{18}	From Last Positive Transition of Transmit Pair Differential Output Approaches Within 40 mV of zero volts.		8000	ns

NOTES:
1. Refer to Figure 11.
2. This parameter is exactly twice t_1.
3. Characterized, not tested.

RECEIVE TIMING(1)

Symbol	Parameter	Min	Max	Unit
t_{19}	Duration which the $\overline{RXC}$ is held at Low State at the Start of a Packet		1400	ns
t_{20}	Receive Pair Signal Rise/Fall Time(5)		10	ns
t_{21}	Receive Pair Bit Cell Center Jitter in Preamble(2)		±12	ns
t_{22}	Receive Pair Bit Cell Center Jitter in Data(2)		±18	ns
t_{23}	Receive Idle Time after Transmission in a Transmitting Station	8		µs
t_{24}	Receive Pair Signal Return to Zero Level from Last Valid Positive Transition	160		ns
t_{25}	$\overline{CRS}$ Assertion Delay from the First Received Valid Negative Transition of Receive Pair Signal		100	ns
t_{26}	$\overline{CRS}$ Deassertion Delay from the Last Valid Positive Transition Received (when no Collision-Presence Signal Exists on the Transceiver Cable)(3)		300	ns
t_{27}	$\overline{RXC}$ Cycle Time	96	104	ns
t_{28}	$\overline{RXC}$ Rise/Fall Time		5.0	ns
t_{29}	$\overline{RXC}$ Low Time	40		ns
t_{30}	$\overline{RXC}$ High Time	36		ns
t_{31}	Receive Data Stable Before the Negative Edge of $\overline{RXC}$	30		ns
t_{32}	Receive Data Held Valid Past the Negative Edge of $\overline{RXC}$	30		ns
t_{33}	Carrier Sense Active → Inactive Hold Time from $\overline{RXC}$ High	10	40	ns
t_{34}	Receive Data Rise/Fall Time(5)		10	ns
t_{35}	$\overline{CRS}$ Inhibit Time After Frame Transmission(4)	5	7	µs

NOTES:
1. Refer to Figures 12 and 13.
2. Measured per 802.3 Para B1.1.4.2 recommendations.
3. $\overline{CRS}$ is deasserted synchronously with the $\overline{RXC}$. This condition is not specified in the IEEE 802.3 specification.
4. Required for SQE test. Applies when $\overline{NOOR}$ = 1. For $\overline{NOOR}$ = 0 there is no inhibit of $\overline{CRS}$.
5. Characterized, not tested.

COLLISION TIMING(1)

Symbol	Parameter	Min	Max	Unit
t_{36}	CLSN/$\overline{\text{CLSN}}$ Cycle Time	80	118	ns
t_{37}	CLSN/$\overline{\text{CLSN}}$ Rise/Fall Time(2)		10	ns
t_{38}	CLSN/$\overline{\text{CLSN}}$ High/Low Time	35	70	ns
t_{39}	CLSN Pair Return to Zero from Last Positive Transition	160		ns
t_{40}	$\overline{\text{CDT}}$ Assertion from the First Valid Negative Edge of Collision Pair Signal		75	ns
t_{41}	$\overline{\text{CDT}}$ Deassertion from the Last Positive Edge of CLSN/$\overline{\text{CLSN}}$ Signal		300	ns
t_{42}	$\overline{\text{CRS}}$ Deassertion from the Last Positive Edge of CLSN/$\overline{\text{CLSN}}$ Signal (if no Post-Collision Signal Remains on the Receive Pair)		450	ns

NOTE:
1. Refer to Figure 14.
2. Characterized, not tested.

LOOPBACK TIMING(1)

Symbol	Parameter	Min	Max	Unit
t_{43}	$\overline{\text{LPBK}}$ asserted before the first attempted transmission (2)	500		ns
t_{44}	Simulated collision test delay from the end of each attempted transmission	0.5	1.5	µs
t_{45}	Simulated collision test duration(3)	0.6	1.6	µs
t_{46}	$\overline{\text{LPBK}}$ deasserted after the last attempted transmission	5		µs

NOTES:
1. Refer to Figure 15.
2. In Loopback mode, $\overline{\text{RXC}}$ and $\overline{\text{CRS}}$ function in the same manner as a normal Receive.
3. SQE test (heartbeat) signal

NOISE FILTER(1)

Symbol	Parameter	Min	Max	Unit
t_{47}	RCV/$\overline{\text{RCV}}$ Noise Filter Pulse Width Rejected		5	ns
t_{48}	RCV/$\overline{\text{RCV}}$ Noise Filter Pulse Width Accepted	30		ns
t_{49}	CLSN/$\overline{\text{CLSN}}$ Noise Filter Pulse Width Rejected		5	ns
t_{50}	CLSN/$\overline{\text{CLSN}}$ Noise Filter Pulse Width Accepted	25		ns

NOTE:
1. Refer to Figure 16.

A.C. TIMING CHARACTERISTICS

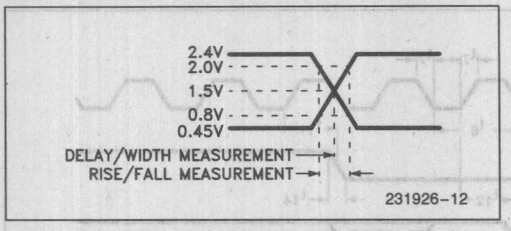


Figure 7. TTL Input Voltage Levels for Timing Measurements ($\overline{TEN}$, TXD, $\overline{LPBK}$/WDTD).

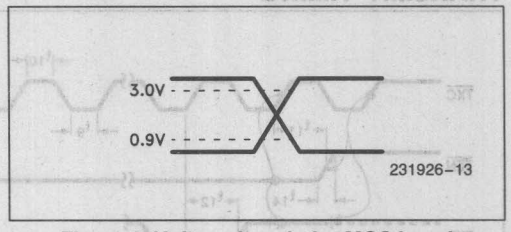


Figure 8. Voltage Levels for MOS Level Output-Timing Measurements ($\overline{TXC}$, $\overline{RXC}$, $\overline{CRS}$, $\overline{CDT}$, and RXD).

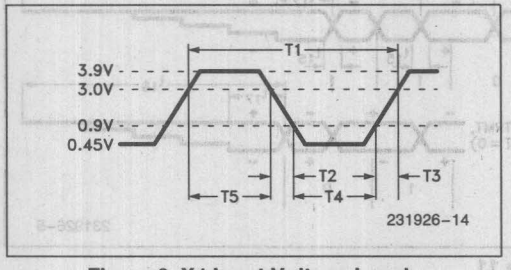


Figure 9. X1 Input Voltage Levels for Timing Measurements

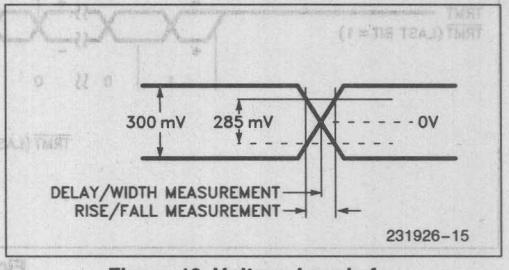


Figure 10. Voltage Levels for Differential-Input Timing Measurements (RCV/$\overline{RCV}$ and CLSN/$\overline{CLSN}$).

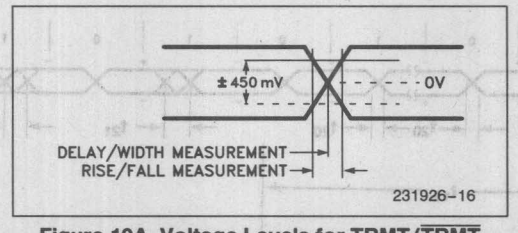


Figure 10A. Voltage Levels for TRMT/$\overline{TRMT}$ Output-Timing Measurements

TRANSMIT TIMING

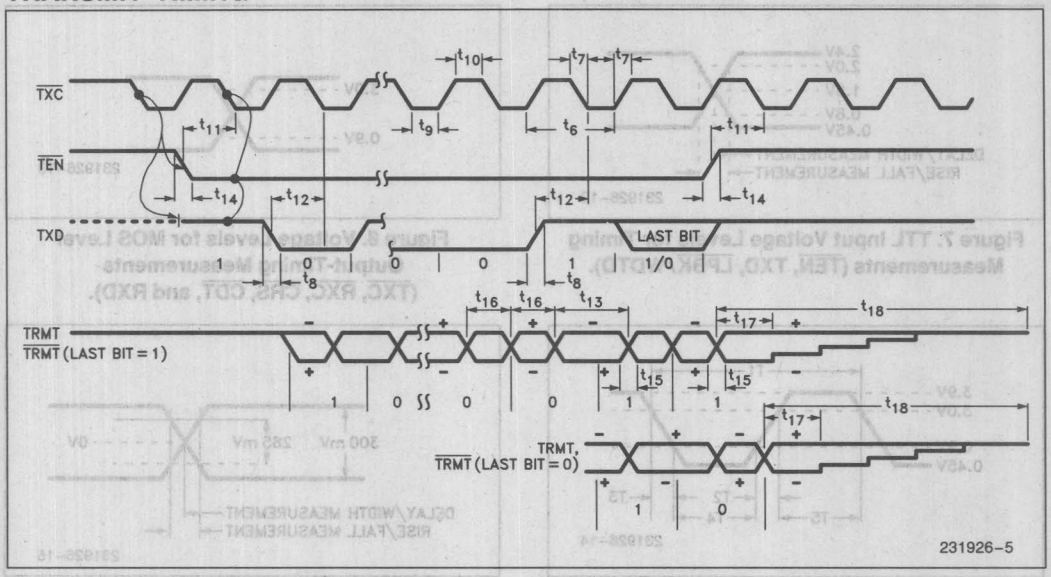


Figure 11

RECEIVE TIMING: START OF FRAME

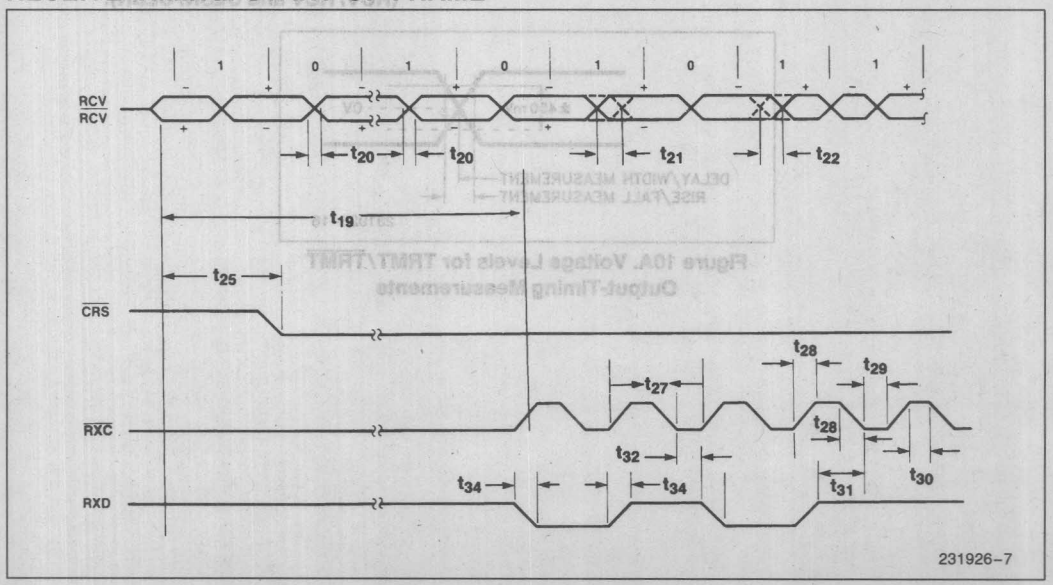


Figure 12

RECEIVE TIMING: END OF FRAME

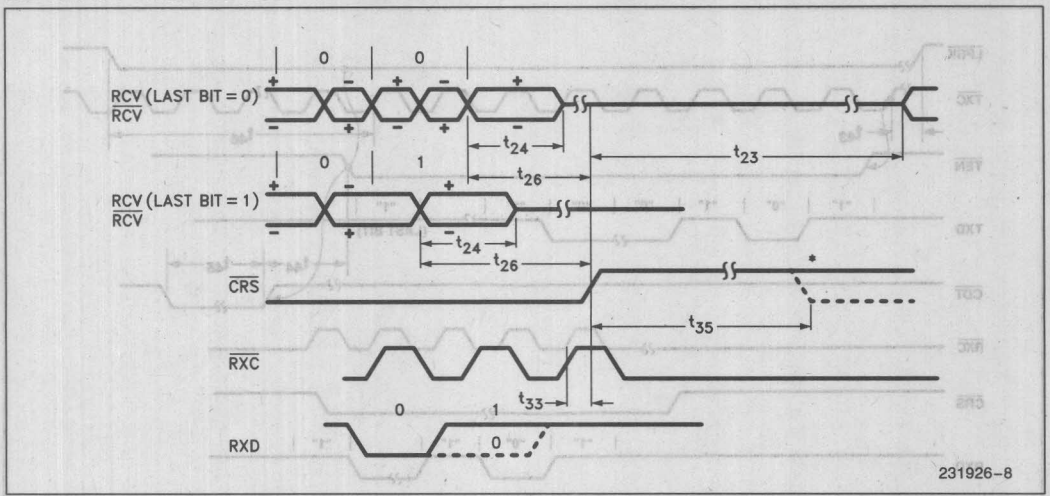


Figure 13

COLLISION TIMING

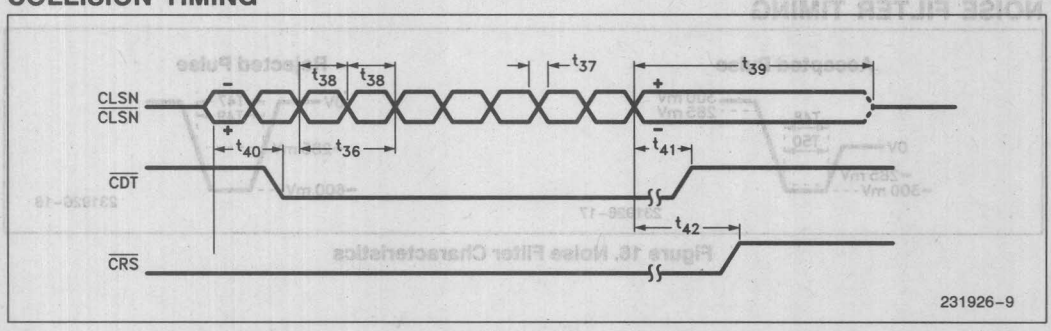


Figure 14

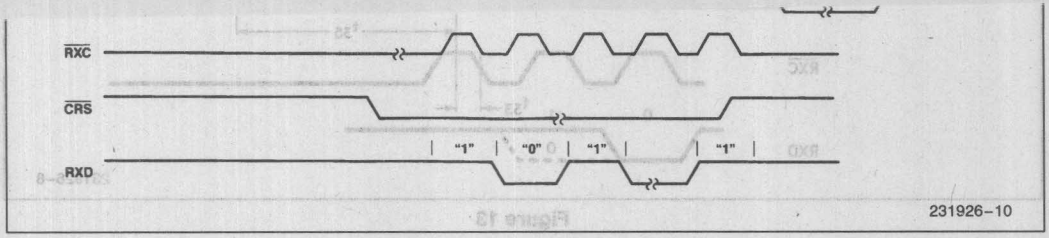


Figure 15

NOISE FILTER TIMING

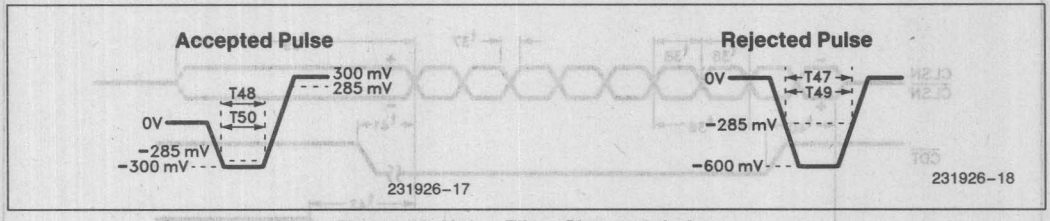
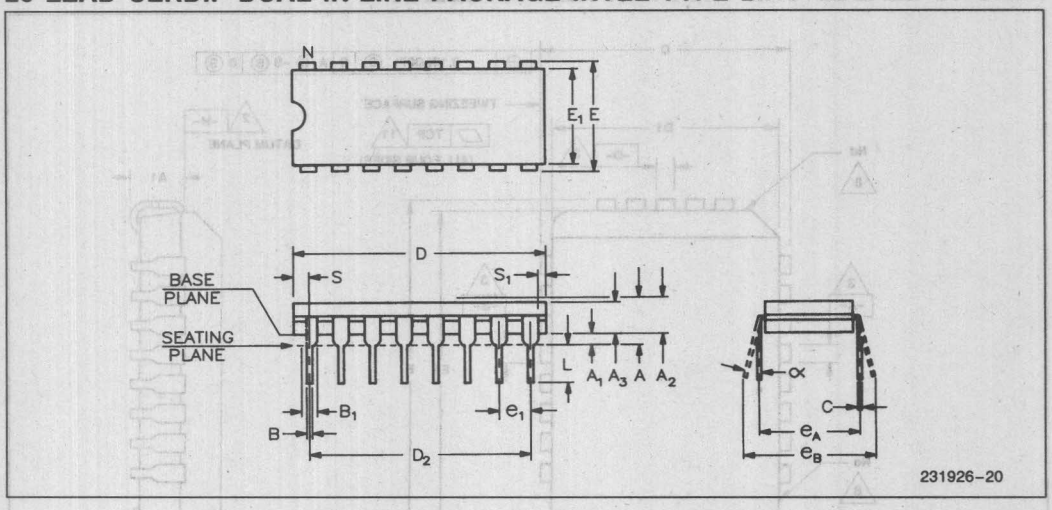


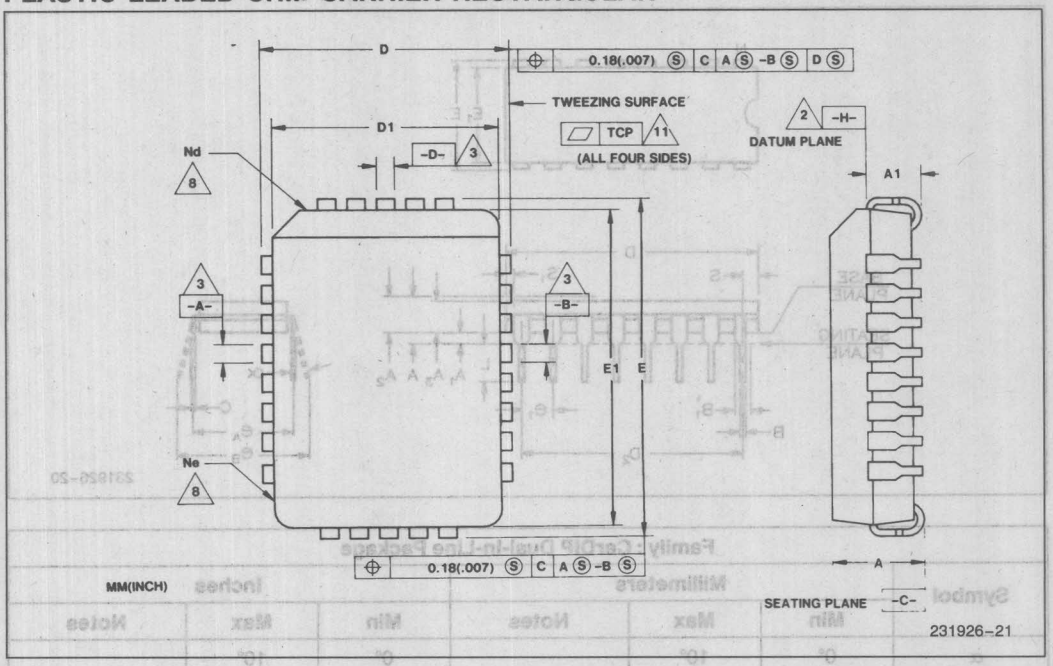
Figure 16. Noise Filter Characteristics

20 LEAD CERDIP DUAL IN-LINE PACKAGE INTEL TYPE D



Family : CerDIP Dual-In-Line Package						
Symbol	Millimeters			Inches		
	Min	Max	Notes	Min	Max	Notes
α	0°	10°		0°	10°	
A		5.08			0.200	
A_1	0.38			0.015		
A_2	3.56	4.24		0.140	0.167	
A_3	3.56	4.24		0.140	0.167	
B	0.41	0.51		0.016	0.020	
B_1	1.52		Typical	0.060		Typical
C	0.23	0.30	Typical	0.009	0.012	Typical
D	24.38	25.27	Reference	0.960	0.995	
D_2	22.86		Reference	0.900		Reference
E	7.62	8.13		0.300	0.320	
E_1	7.11	7.90		0.280	0.31	
e_1	2.29	2.79		0.090	0.110	
e_A	7.87		Reference	0.310		Reference
e_B	8.13	10.16		0.320	0.400	
L	3.18	3.81		0.125	0.150	
N	20		½ Leads	20		½ Leads
S	0.38	1.78		0.015	0.070	
S_1	0.13			0.005		
ISSUE	IWS 1/15/87					

PLASTIC LEADED CHIP CARRIER RECTANGULAR



Dimension	28 Lead (Inch)		28 Lead (mm)	
	Min	Max	Min	Max
Overall Height (A)	0.126	0.140	3.20	3.56
Shoulder to Board Height (A_1)	0.076	0.090	1.93	2.29
Outside Dimension (D)	0.385	0.396	9.78	10.0
Plastic Body Dimension (D_1)	0.347	0.353	8.81	8.97
Foot Print (D_2)	0.290	0.330	7.37	8.38
Foot Print (D_3)	0.200 Ref.		5.08 Ref.	
Outside Dimension (E)	0.585	0.595	14.9	15.1
Platic Body Dimension (E_1)	0.547	0.553	13.9	14.0
Foot Print (E_2)	0.490	0.530	12.4	13.5
Foot Print (E_3)	0.400 Ref.		10.2 Ref.	
# of Leads (N)	28		28	
# of Leads on Short Side (N_d)	5		5	
# of Leads on Long Side (N_e)	9		9	
Seating Plane Coplanarity (CP)	0.000	0.004	0.00	0.10
Tweezing Coplanarity (TCP)	0.000	0.004	0.000	0.10
Lead Thickness (LT)	0.009	0.015	0.23	0.38
Issue	IWS 1/15/87			

82521TB
TWISTED PAIR ETHERNET* SERIAL SUPERCOMPONENT

- **Provides Complete Serial and Analog Twisted Pair Ethernet Interface**
  - **— Analog Filters**
  - **— Serial Interface and Transceiver**
  - **— Manchester Encoder and Decoder**
  - **— Link Integrity**
  - **— Line Drivers and Receivers**
  - **— Jabber Protection**
  - **— Isolation Transformers and Protection Circuitry**
- **Designed to IEEE 802.3 Draft Supplement, Type 10BASE-T (P802.3I/D10)**
- **10-Mb/s Operation**
- **Directly Interfaces Intel Ethernet LAN Controllers and Coprocessors**
  - **— 82586**
  - **— 82590 and 82592**
  - **— 82596CA, 82596DX, and 82596SX**
- **Socket Compatible with the 82521TA**
- **No Configuration Required**
- **Allows Design to Meet FCC Class A Standard**
- **All Circuitry in a Single 36-Pin Package**

1

The Twisted Pair Ethernet Serial Supercomponent (SSC) provides the complete serial and analog Twisted Pair Ethernet interface required to connect the Ethernet LAN controller directly to a 10BASE-T connector. It is designed for node applications in 10-Mb/s, CSMA/CD networks as defined by the IEEE 802.3-1985 standard; for example, PCs, workstations, and fileservers. The SSC includes the serial interface, transceiver, Manchester encoder, Manchester decoder, 10BASE-T functionality, line drivers, line receivers, analog filters, protection circuitry and isolation transformers in a single package. It provides all the required circuitry to give the LAN designer immediate access to the twisted pair Ethernet environment (10BASE-T).

Existing Ethernet/Cheapernet designs can be easily modified to take advantage of cost-effective twisted pair wire. The SSC can be soldered or socketed onto a host adapter card or motherboard without any adjustments or configuration. It is compatible with the pending IEEE 802.3 draft standard 10BASE-T, and can be used with non-Intel LAN controllers. The SSC is designed to satisfy FCC class A test requirements and to meet all standard host-system size and power requirements.

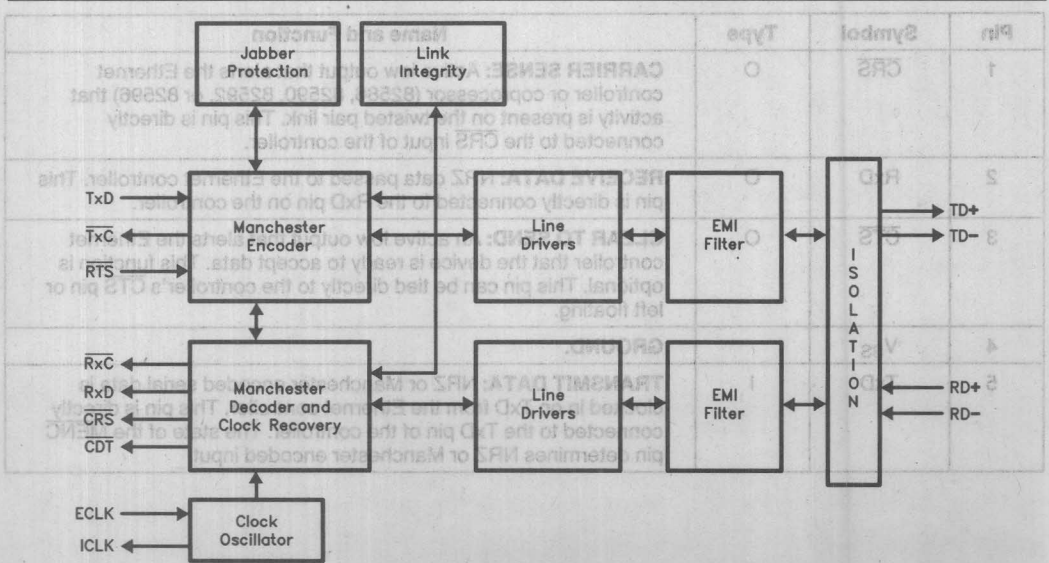
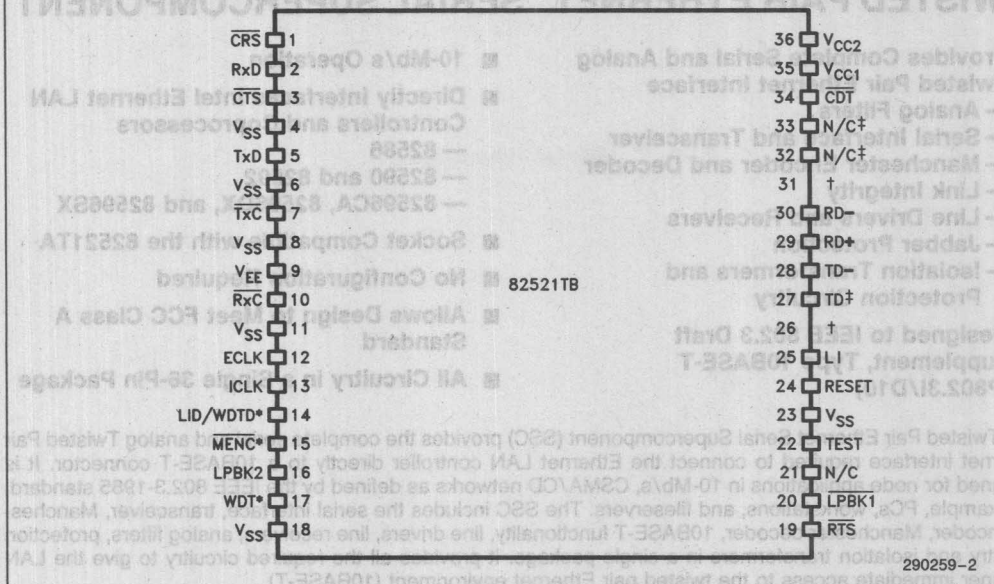


Figure 1. 82521TB Block Diagram

*Ethernet is a registered trademark of Xerox Corporation.

September 1990
Order Number: 290259-003



*Pins 14, 15, and 17 have been enhanced since the 82521TA. This does not affect compatibility with the 82521TA.
†Pins 26 and 31 have been removed for improved electrical isolation. This does not affect compatibility with the 82521TA.
‡The 12-V power requirement of the 82521 has been removed on the 82521TB. To maintain socket compatibility these pins are not connected internally; they can be connected to a power supply on the host card or left floating.

Figure 2. 82521TB Pin Configuration

Pin	Symbol	Type	Name and Function
1	$\overline{CRS}$	O	**CARRIER SENSE:** Active low output that alerts the Ethernet controller or coprocessor (82586, 82590, 82592, or 82596) that activity is present on the twisted pair link. This pin is directly connected to the $\overline{CRS}$ input of the controller.
2	RxD	O	**RECEIVE DATA:** NRZ data passed to the Ethernet controller. This pin is directly connected to the RxD pin on the controller.
3	$\overline{CTS}$	O	**CLEAR TO SEND:** An active low output that alerts the Ethernet controller that the device is ready to accept data. This function is optional. This pin can be tied directly to the controller's $\overline{CTS}$ pin or left floating.
4	V_{SS}		**GROUND.**
5	TxD	I	**TRANSMIT DATA:** NRZ or Manchester encoded serial data is clocked in on TxD from the Ethernet controller. This pin is directly connected to the TxD pin of the controller. The state of the $\overline{MENC}$ pin determines NRZ or Manchester encoded input.

Pin	Symbol	Type	Name and Function
6	V_{SS}		**GROUND.**
7	$\overline{TxC}$	O	**TRANSMIT CLOCK:** A 10-MHz clock output tied directly to the $\overline{TxC}$ pin of the Ethernet controller.
8	V_{SS}		**GROUND.**
9	V_{EE}		**V_{EE}:** A −5 V ±10% power supply.
10	$\overline{RxC}$	O	**RECEIVE CLOCK:** A 10-MHz clock connected directly to the $\overline{RxC}$ input of the Ethernet controller. This clock is the recovered clock from the incoming data on the twisted pair.
11	V_{SS}		**GROUND.**
12	ECLK	O	**EXTERNAL CLOCK:** A 20-MHz ±0.01%, TTL-level input clock with a 40/60 duty cycle. ECLK attaches to ICLK for normal asynchronous operation. Synchronous operation of the device can be obtained by connecting an external clock to ECLK.
13	ICLK	I	**INTERNAL CLOCK:** A 20-MHz ±0.01%, TTL-level output clock with a 40/60 duty cycle. For normal asynchronous operation this pin is directly connected to ECLK. ICLK can be used for synchronous operation of interface circuitry.
14	LID/WDTD	I	**LINK INTEGRITY DISABLE, WATCHDOG TIMER DISABLE:** When connected to V_{CC1}, this pin disables the link integrity processor, linkbeat generator, and watchdog timer. This ensures compatibility with the 82521TA. When connected to V_{SS}, or not connected, these functions are enabled and the device is compatible with 10BASE-T.
15	$\overline{MENC}$	I	**MANCHESTER ENCODING:** When tied to V_{SS}, or not connected, this pin enables internal Manchester encoding of the NRZ data on TxD. When tied to V_{CC1}, Manchester encoded data is expected on TxD.
16	LPBK2	I	**LOOPBACK 2:** An active high input signal that causes the 82521TB to enter diagnostic loopback mode. The twisted pair medium will be removed from the circuit, thus isolating the node from the network. When not connected this pin assumes the inactive (normal) state. Diagnostic loopback mode does not affect the operation of the link integrity processor or linkbeat generator. The watchdog timer will not operate in diagnostic loopback mode.
17	WDT	O	**WATCHDOG TIMER:** An active high output that indicates expiration of the watchdog timer (jabber protection) in the super-component. The output deasserts when the jabber function is reset.
18	V_{SS}		**GROUND.**
19	$\overline{RTS}$	I	**REQUEST TO SEND:** An active low input signal synchronous to $\overline{TxC}$; it enables data transmission on the twisted pair link segment.
20	$\overline{LPBK1}$	I	**LOOPBACK 1:** An active low input signal that causes the 82521TB to enter diagnostic loopback mode. The twisted pair medium will be removed from the circuit, thus isolating the node from the network. When not connected this pin assumes the inactive (normal) state. Diagnostic loopback mode does not affect the operation of the link integrity processor, or linkbeat generator. The watchdog timer will not operate in diagnostic loopback mode.

Pin	Symbol	Type	Name and Function
21	N/C		This pin is not connected.
22	Reserved	I	This pin is used for testing purposes. It should be left floating or connected to V_{SS}.
23	V_{SS}		**GROUND.**
24	Reset	I	**RESET:** An active high input that brings the device into a known state. It must be asserted for 1 ms while the clock is running.
25	LI	O	**LINK INTEGRITY:** An active high output used to indicate the presence of link integrity faults.
27	TD+	O	**TWISTED PAIR TRANSMIT DATA:** This pin transmits outgoing Manchester data to the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 1. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.
28	TD−	O	**TWISTED PAIR TRANSMIT DATA COMPLEMENT:** This pin transmits outgoing inverted Manchester data to the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 2. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.
29	RD+	I	**TWISTED PAIR RECEIVE DATA:** This pin receives incoming Manchester data from the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 3. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.
30	RD−	I	**TWISTED PAIR RECEIVE DATA COMPLEMENT:** This pin receives incoming inverted Manchester data from the twisted pair link segment. It is connected directly to the Medium Dependent Interface Connector (RJ-45) pin 6. This pin, and the trace leading to it, must withstand 2250 V dc to ground without damage.
32	N/C		This pin is not connected.
33	N/C		This pin is not connected.
34	$\overline{CDT}$	O	**COLLISION DETECT:** An active low signal that indicates the presence of a collision to the controller.
35	V_{CC1}		**V_{CC1}:** A 5 V ±5% power supply.
36	V_{CC2}		**V_{CC2}:** A 5 V ±10% power supply. This pin can be connected to pin 35 (V_{CC1}) if a single power supply can reliably supply the combined requirements of I_{CC1} and I_{CC2} (see DC Characteristics).

FUNCTIONAL DESCRIPTION

Overview

The 82521TB provides the functions required for operating Data Terminal Equipment on a 10-Mb/s, CSMA/CD, Local Area Network (LAN) using standard telephone building wiring. The 82521TB design is based on the Twisted Pair Ethernet draft standard supplement to IEEE Std. 802.3 (type 10BASE-T, P802.3I/D8).

Clock Generation

The 82521TB supports internal and external sources for the precision clock (20 MHz ±0.01%) required in an Ethernet environment. The clock is used to retime the transmitted Manchester data, to generate the 10-MHz $\overline{\text{TxC}}$ signal, and as a precision reference for Manchester decoding and clock recovery of received data.

If ICLK (Pin 13) is strapped to ECLK (Pin 12) an onboard clock oscillator generates the precision clock. This clock can be used for synchronous operation of circuits on the host board; however, care must be taken to minimize the load on the clock. The supercomponent can be operated synchronous to the host by providing ECLK with a 20-MHz ±0.01%, TTL-level clock with a duty cycle of 40/60 or better.

Transmit Section

The transmit section of the 82521TB is controlled by the $\overline{\text{RTS}}$ signal generated by the Ethernet LAN controller (Intel's 82586, 82592, etc). When $\overline{\text{RTS}}$ asserts, the 82521TB begins clocking in data from the controller on the TxD input. The TxD level is sampled on the falling edge of the 10-MHz $\overline{\text{TxC}}$ signal (or every edge if Manchester encoding is not enabled). The data is then encoded using the precision 20-MHz clock, and then sent to the twisted pair line drivers.

The line drivers begin transmitting the serial Manchester bit stream two bit times after the assertion of $\overline{\text{RTS}}$. A predistortion algorithm is used by the line drivers to improve jitter performance on the twisted pair. The line drivers reduce their drive level during the second half of "fat" (100 ns) Manchester pulses, and maintain full drive level during all "thin" (50 ns) pulses. During the "fat" pulses, this reduces line overcharge, which is a major source of jitter. Figure 3 shows the difference between the familiar coax cable waveform and the predistorted waveform generated by the 82521TB. The line drivers maintain a characteristic impedance of 96 Ω typical throughout data packet transmission and the idle state.

The predistorted output of the line drivers is then passed through the transmit EMI filter to reduce the high frequency harmonics of the transmitted signal. This reduces noise, Near End Cross Talk (NEXT) in bundled twisted pair cables, and unwanted Radio Frequency (RF) interference. The filter maintains the 96 Ω (typical) characteristic impedance. The filtered signal then passes through an isolation transformer and common mode choke. These provide high voltage protection, dc isolation, and common mode noise rejection. The output of the common mode choke is directly connected to the TD+ and TD− pins. These can be connected to pins 1 and 2 of an ISO 8877 (RJ-45) connector.

After a successful transmission, the *signal_quality_error* test (heartbeat) function is executed in accordance with the 802.3 10BASE-T draft specification.

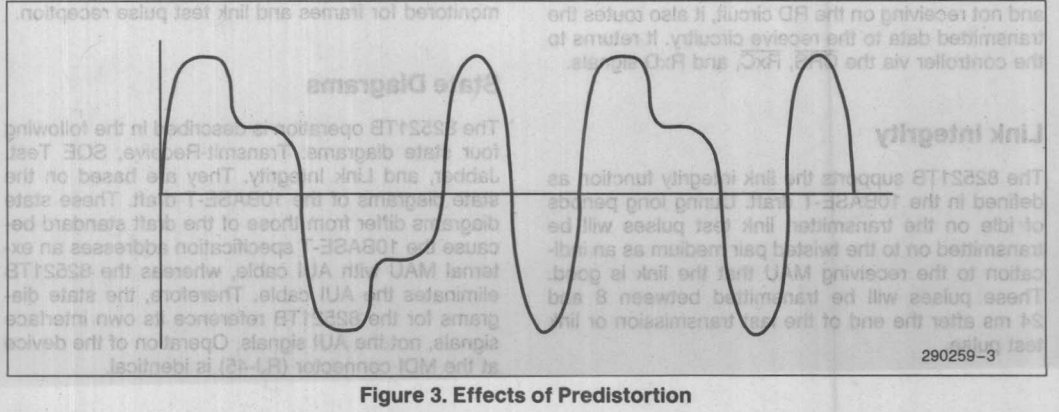


Figure 3. Effects of Predistortion

Receive Section

The receive section of the 82521TB processes incoming Manchester data from the twisted pair link segment, converts it to NRZ data, and recovers the embedded clock. It contains a squelch circuit that distinguishes noise from incoming data. Valid data passes through the input protection and common mode rejection of the 82521TB and the receive EMI filter, and trips the squelch circuit. The twisted pair line receiver is then enabled, and converts the signal to digital voltage levels.

The signal then passes to the Manchester decoder and clock recovery circuit. $\overline{CRS}$ asserts within nine bit times of the arrival of the data packet to indicate the presence of activity on the network. Fourteen bit times later $\overline{RxC}$ and RxD activate, passing the remaining preamble and data to the controller in NRZ format.

The 82521TB detects the Start-Of-Idle (SOI) signal at the end of a packet. $\overline{CRS}$ will be synchronously deasserted with $\overline{RxC}$ within four bit times from the beginning of the SOI. $\overline{RxC}$ and RxD then return to their idle state.

Collision Detect

Collision detection in the twisted pair environment is indicated by simultaneous transmission and reception on the twisted pair link segment. The $\overline{CDT}$ signal is asserted for the duration of both $\overline{RTS}$ and the presence of received data; $\overline{CRS}$ is asserted for the duration of either $\overline{RTS}$ or the presence of received data.

DO to DI Loopback

When the 82521TB is transmitting on the TD circuit, and not receiving on the RD circuit, it also routes the transmitted data to the receive circuitry. It returns to the controller via the $\overline{CRS}$, $\overline{RxC}$, and RxD signals.

Link Integrity

The 82521TB supports the link integrity function as defined in the 10BASE-T draft. During long periods of idle on the transmitter, link test pulses will be transmitted on to the twisted pair medium as an indication to the receiving MAU that the link is good. These pulses will be transmitted between 8 and 24 ms after the end of the last transmission or link test pulse.

The link integrity function continuously monitors activity on the receive circuit. If neither valid data or link test pulses are received for a period of time, the link integrity processor declares the link bad, and disables transmission and reception on the medium. Transmission of link test pulses and monitoring of receive activity are not affected. The idle time required for the link integrity processor to determine if the link is bad is between 50 and 150 ms.

Once a frame, or a sequence of 2 to 10 consecutive link test pulses, are detected, the link integrity processor declares the link good and reconnects the transmitter and receiver.

Jabber Function

The 82521TB has an onboard watchdog timer to implement the jabber function. If a transmission continues beyond the limits specified by the 10BASE-T draft standard (between 20 and 150 ms), the jabber function inhibits further transmission and asserts the collision indicator $\overline{CDT}$. The transmission inhibit period extends until the 82521TB detects sufficient idle time (between 250 and 750 ms) on the $\overline{RTS}$ signal. Link test pulses continue to be sent during the period when the transmitter is disabled.

Diagnostic Loopback Mode

The 82521TB supports a diagnostic loopback mode in addition to the normal DO to DI loopback mode. When either $\overline{LPBK1}$ or LPBK2 are asserted, data transmission and reception on the twisted pair link is disabled, thus removing the DTE from the network. Any transmissions made in this mode are fed back into the receive circuits and subsequently passed back to the controller. Diagnostic loopback mode does not affect the link integrity function. Link test pulses are still transmitted and the twisted pair link is monitored for frames and link test pulse reception.

State Diagrams

The 82521TB operation is described in the following four state diagrams: Transmit-Receive, SQE Test, Jabber, and Link Integrity. They are based on the state diagrams of the 10BASE-T draft. These state diagrams differ from those of the draft standard because the 10BASE-T specification addresses an external MAU with AUI cable, whereas the 82521TB eliminates the AUI cable. Therefore, the state diagrams for the 82521TB reference its own interface signals, not the AUI signals. Operation of the device at the MDI connector (RJ-45) is identical.

FUNCTIONAL DESCRIPTION (Continued)

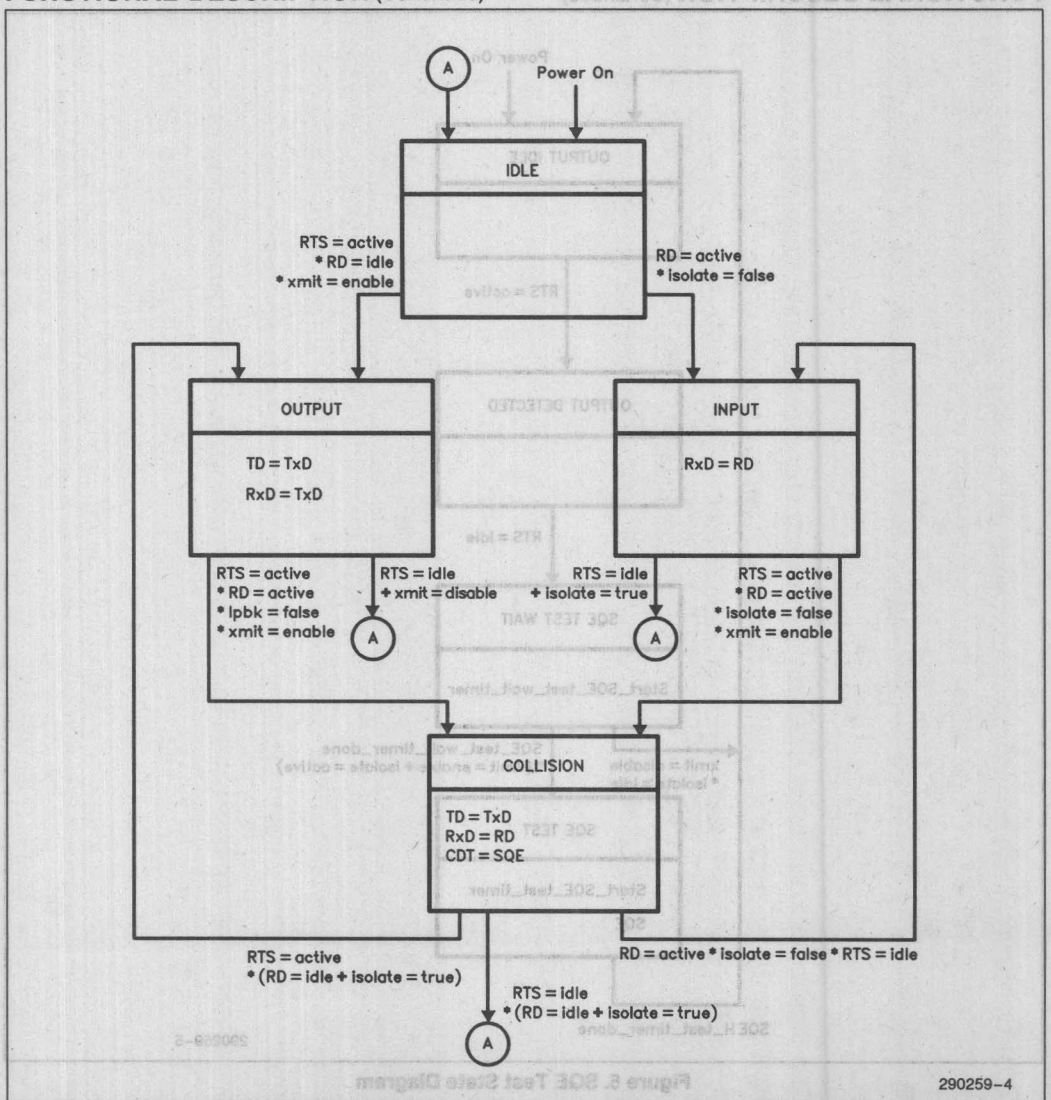


Figure 4. Transmit Receive State Diagram

FUNCTIONAL DESCRIPTION (Continued)

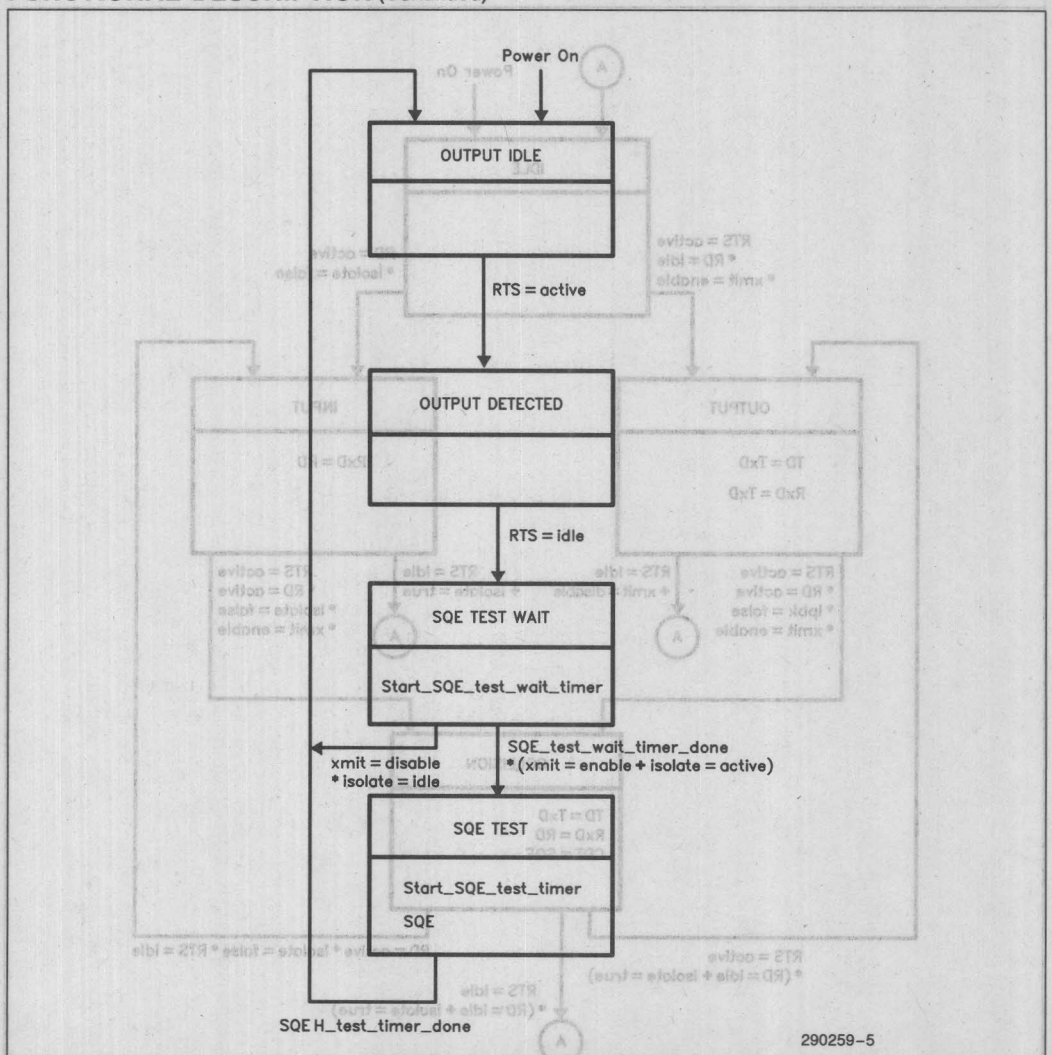


Figure 5. SQE Test State Diagram

FUNCTIONAL DESCRIPTION (Continued)

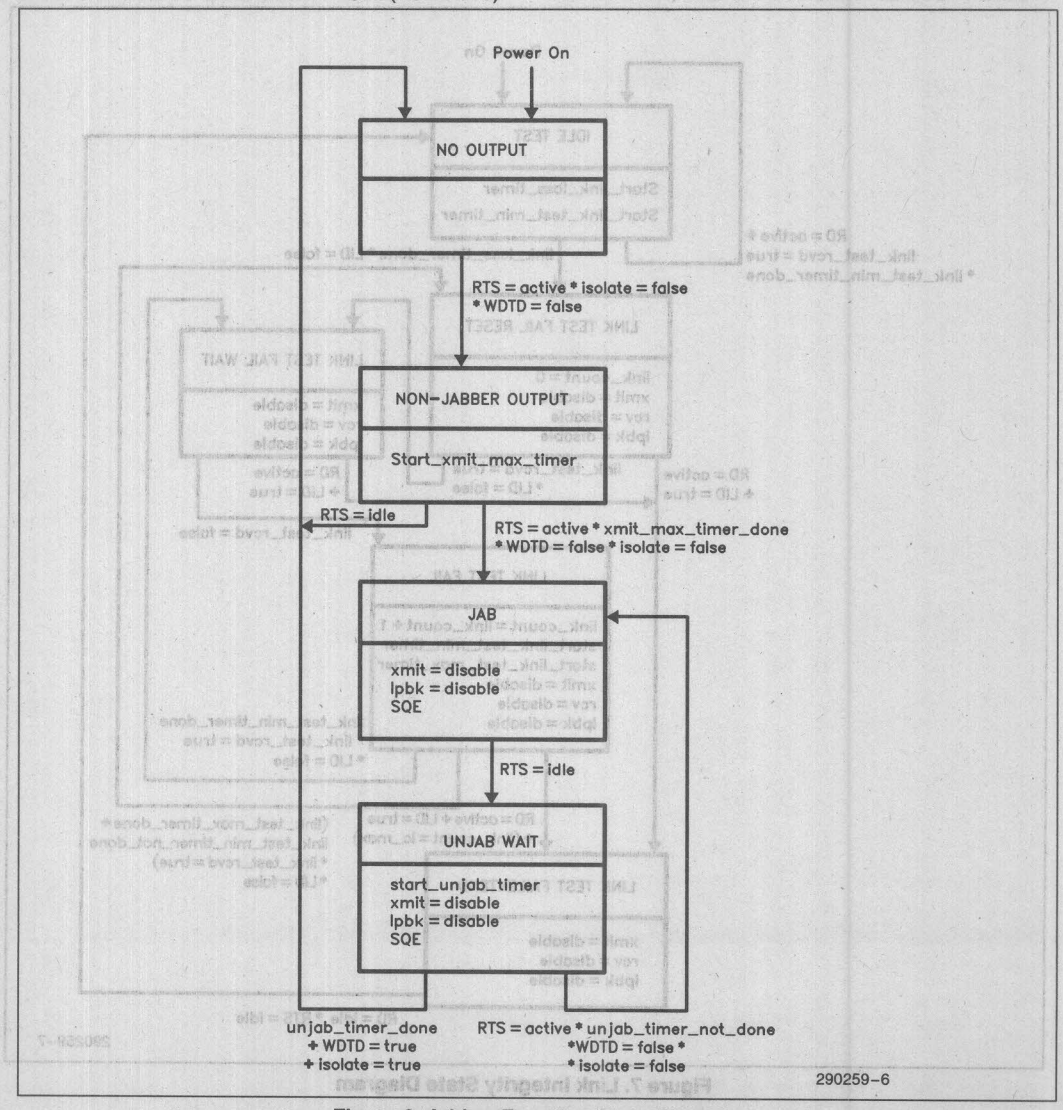


Figure 6. Jabber Function State Diagram

FUNCTIONAL DESCRIPTION (Continued)

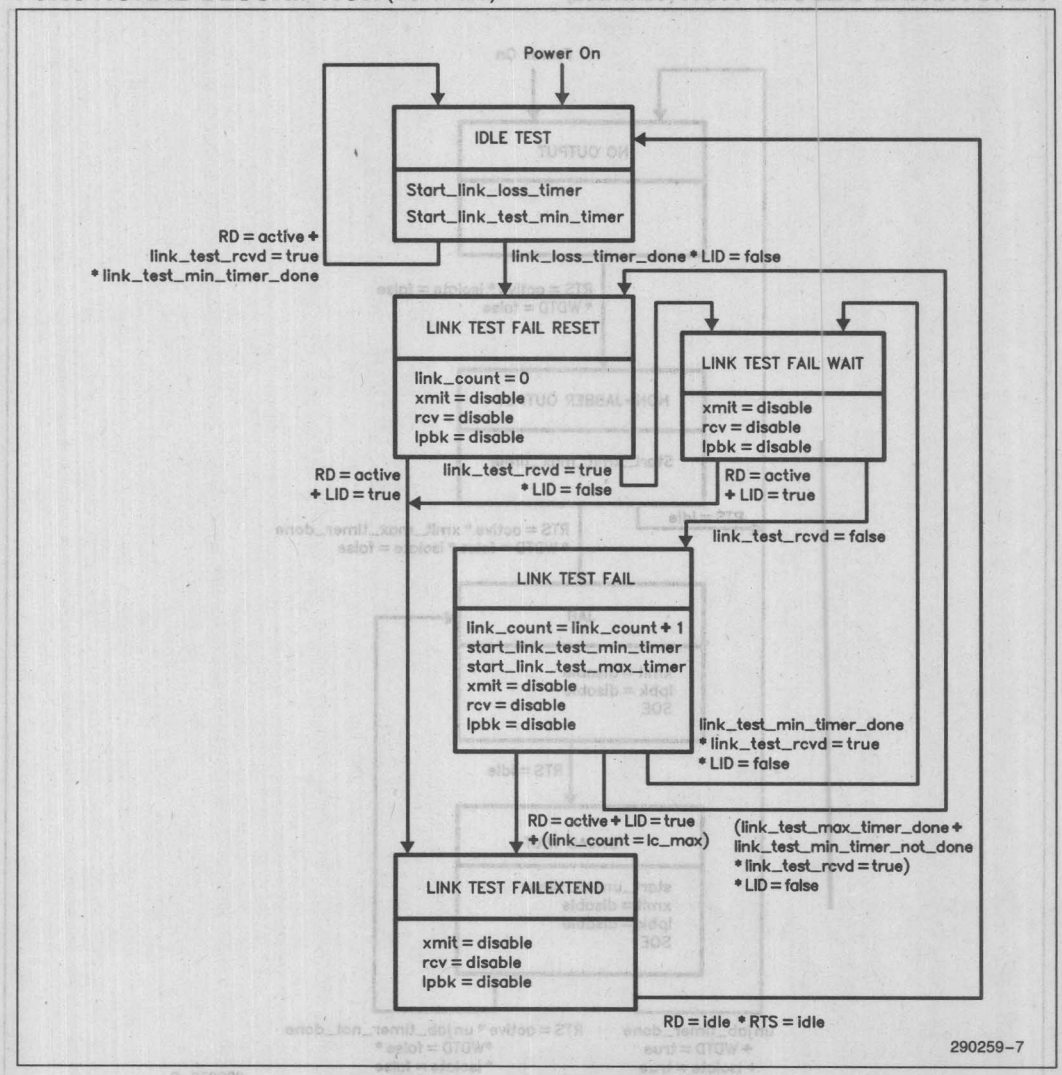


Figure 7. Link Integrity State Diagram

State Diagram Variables

The variable definitions contain a description of the function they control as well as the possible values for that variable. Many of the variables have a default value; when none of the state machines explicitly assigns a particular variable a value, then that variable implicitly takes on the default value. Several constants are used and defined by either the 802.3 standard or the 10BASE-T draft, these constants are printed italic.

RTS Indicates the presence of a data stream being received from the DTE on the TxD signal.

RTS = active. Data is present on the TxD circuit.

RTS = idle. Data is not present on the TxD circuit.

RxD Controls the source of data transmitted to the DTE on the RxD circuit.

RxD = RD. Data on the RxD circuit is sourced by the RD circuit input from the twisted pair. The condition "rcv = disable" overrides and disables this activity.

RxD = DO. Data on the RxD circuit is sourced by the TxD circuit from the DTE. This is the loopback function. The condition "loopback = disable" overrides and disables this activity.

RxD = idle. Default. There is no data being transmitted on the RxD circuit, it contains the *input_idle* message.

CDT Controls the message the 82521TB sends to the DTE on the $\overline{\text{CDT}}$ signal.

CDT = SQE. The MAU is sending the *signal_quality_error* message to the DTE on the $\overline{\text{CDT}}$ signal.

CDT = idle. Default. the MAU is sending the *MAU_available* message to the DTE on the $\overline{\text{CDT}}$ signal.

TD Controls the source of data transmitted to the network on the TD circuit.

TD = TxD. The TD circuit is sourced by the TxD signal. The conditions "xmit = disable" or "isolate = true" overrides and disables this activity.

RD Indicates the presence of an input data stream from the network on the RD circuit.

RD = active. Data is present on the RD circuit.

RD = idle. Data is not present on the RD circuit.

link_test_rcvd Indicates the presence of a link test pulse on the RD circuit.

link_test_rcvd = true. A link test pulse is present on the RD circuit.

link_test_rcvd = false. Silence or data is present on the RD circuit.

link_count Indicates the number of consecutive link test pulses received while in a link-fail-state.

lc_max The maximum number of consecutive link test pulses required before reconnection is allowed.

xmit Communication path between state machines. This variable relates the status of certain fault conditions that require the transmit and SQE functions to be disabled.

xmit = disable. A condition exists that dictates the transmit and SQE test functions to be disabled.

xmit = enable. Default. The transmit and SQE test functions operate normally.

rcv Communication path between state machines. This variable relates the status of certain fault conditions that require the receive function to be disabled.

rcv = disable. A condition exists that dictates that the receive function be disabled.

rcv = enable. Default. The receive function should operate normally.

lpbk Communication path between state machines. This variable relates the status of certain fault conditions that require the DO to DI loopback function to be disabled.

lpbk = disable. A condition exists that dictates that the loopback function be disabled.

lpbk (continued)

lpbk = enable. Default. The DO to DI loopback function should operate normally.

isolate

Indicates the status of the diagnostic loopback mode of the 82521TB.

isolate = false. Device is not in diagnostic loopback mode. Input $\overline{\text{LPBK1}}$ = 1 and LPBK2 = 0.

isolate = true. Device is in diagnostic loopback mode. Input $\overline{\text{LPBK1}}$ = 0 or LPBK2 = 1.

WDTD

Indicates the status of the watchdog timer disable mode of the 82521TB.

WDTD = true. Watchdog timer (Jabber function) is disabled. Input LID/WDTD = 1.

WDTD = false. Watchdog timer is enabled. Input LID/WDTD = 0.

LID

Indicates the status of the link integrity disable mode of the 82521TB.

LID = true. Link integrity process is disabled. Input LID/WDTD = 1.

LID = false. Link integrity process is enabled. Input LID/WDTD = 0.

State Diagram Timers

link_loss_timer. Time to wait to declare a bad receive link.

link_test_min_timer. Minimum time allowed between consecutive link test pulses.

link_test_max_timer. Maximum time allowed between consecutive link pulses.

SQE_test_wait_timer. Time to wait before executing the SQE_test function.

SQE_test_timer. Time to wait for completing the SQE_test function.

xmit_max_timer. Time to wait to interrupt jabbering transmission.

unjab_timer. Time to wait before resetting jabber function.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature Under Bias .....0° to +70°C
Storage Temperature .....−65° to +140°C
All Output and Supply Voltages(1) ....−0.5 to +7V
All Input Voltages(2) .....−0.5 to V_{CC1} +0.5V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0° to 70°C, V_{CC1} = 5 V ±5%, V_{CC2} = 5 V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (3)	−0.3	0.8	V	
V_{IH}	Input High Voltage (3)	2.0	V_{CC}	V	
I_{LI}	Input Leakage Current		±400	μA	$-0.3\ V \le V_I \le V_{CC1}$
V_{OL}	Output Low Voltage (4)		0.45	V	I_{OL} = 4 mA
V_{OH}	Output High Voltage (4)	3.9		V	I_{OH} = −500 μA
I_{CC1}	Power Supply Current		410	mA	V_{CC} = 5.25 V
I_{CC2}	Power Supply Current		105	mA	V_{CC} = 5.5 V
I_{EE}	Power Supply Current		−40	mA	V_{EE} = −5.5 V
PD	Power Dissipation		3	W	V_{CC} = 5.0 V

NOTES:
1. The voltage levels for TD+ and TD− are ±2250 V with respect to ground.
2. The voltage levels for RD+ and RD− are ±2250 V with respect to ground.
3. Digital Inputs, TxD, ECLK, $\overline{\text{MENC}}$, LID/$\overline{\text{WDTD}}$, LPBK2, $\overline{\text{RTS}}$, $\overline{\text{LPBK1}}$, TEST, and RESET.
4. Digital Outputs, $\overline{\text{CRS}}$, RxD, $\overline{\text{CTS}}$, $\overline{\text{TxC}}$, $\overline{\text{RxC}}$, ICLK, WDT, LI, and $\overline{\text{CDT}}$.

ANALOG CHARACTERISTICS T_A = 0° to 70°C, V_{CC1} = 5 V ±5%, V_{CC2} = 5 V ±10%

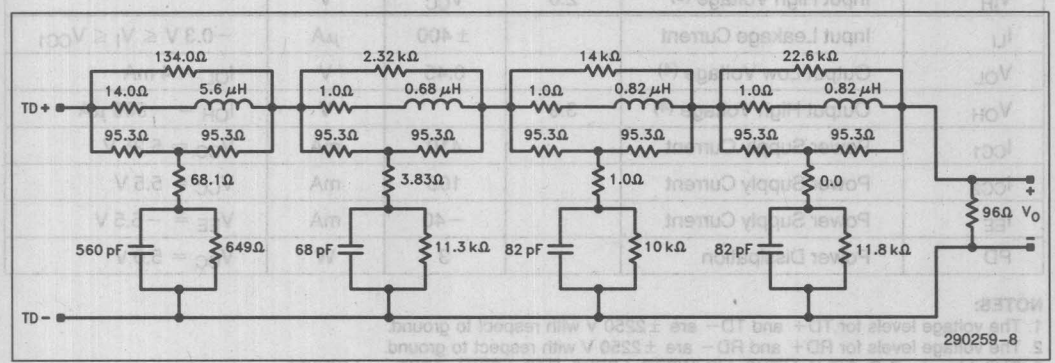
Symbol	Parameter	Min	Max	Units	Test Conditions
Z_O	Characteristic Impedance(1, 2)	77	115	Ω	5 MHz to 10 MHz
	Return Loss(1, 2) (5 MHz to 10 MHz)	15		dB	$85\Omega \le R_{LOAD} \le 111\Omega$
	Squelch Reject Level(2)		300	mV	
	Squelch Accept Level(2)	450		mV	
CMR	Common Mode Rejection(1, 2)	29		dB	At 10 MHz
V_{IDF}	Input Peak Differential	0.500		V	
V_{ODF1}	Output Peak Differential	2.2	2.8	V	96Ω Load

ANALOG CHARACTERISTICS (Continued)

T_A = 0° to 70°C, V_{CC1} = 5 V ±5%, V_{CC2} = 5 V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{ODF2}	Output Peak Differential	0.585		V	With Cable Model(3)
	Maximum Overshoot		50	mV	Start of Idle
	Transmitter Impedance Balance	29		dB	At 10 MHz
V_{OCM}	Output Common Mode		50	mV	>30 kHz
	Harmonic Content		−27	dB	≥30 MHz

NOTES:
1. Pin 27 and 28 (TD+ and TD−).
2. Pins 29 and 30 (RD+ and RD−).
3. Cable Mode defined in 10BASE-T draft.



A.C. Timing Conditions

1. T_A = 0° to 70°C, V_{CC1} = 5 V ±5%, V_{CC2} = 5 V ±10%.
2. Digitial outputs timing measurement points are 0.9 V and 3.0 V unless otherwise noted.
3. Digitial input timing measurements points are 0.8 V and 2.0 V unless otherwise noted.
4. TD Pair and RD Pair timing measurements are 0 V unless otherwise noted.
5. Digital ac loads: 20 pF to ground.
6. TD+ and TD− load: 96 Ω differential load.

CLOCK TIMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
t_1	ECLK/ICLK Cycle Time	49.995	50.005	ns
t_2	ECLK/ICLK Rise and Fall Time		8	ns
t_3	ECLK/ICLK High and Low Time	15		ns

Decoder Timing Characteristics

Symbol	Parameter	Min	Max	Units
t_4	$\overline{RxC}$ Period	96	104	ns
t_5	$\overline{RxC}$ High Time	36		ns
t_6	$\overline{RxC}$ Low Time	40		ns
t_7	$\overline{RxC}$ Rise and Fall Time(2)		5	ns
t_8	RxD Setup Time to $\overline{RxC}$ Falling Edge	30		ns
t_9	RxD Hold Time from $\overline{RxC}$ Falling Edge	30		ns
t_{10}	RxD Rise and Fall Time(2)		5	ns
t_{11}	Receiver Steady State Delay		300	ns
t_{12}	$\overline{CRS}$ Assertion Delay from First Valid RD Pair Edge(1)		600	ns
t_{13}	Duration $\overline{RxC}$ is Held Low at Start of Packet		1900	ns
t_{14}	$\overline{CRS}$ Deassertion Delay from Last Valid RD Pair Edge(1)		350	ns
t_{15}	$\overline{CRS}$ Deassertion Hold Time from $\overline{RxC}$ High(1)	10	40	ns
t_{16}	RD Pair Bit Cell Center Jitter in Preamble		±12	ns
t_{17}	RD Pair Bit Cell Center Jitter in Data		±18	ns
t_{18}	RD Pair Return to Zero from Last Valid Positive Transition	235		ns
t_{19}	RD Idle Time After Transmission	8		μs

NOTES:
1. $\overline{RTS}$ inactive.
2. Characterized, not tested.

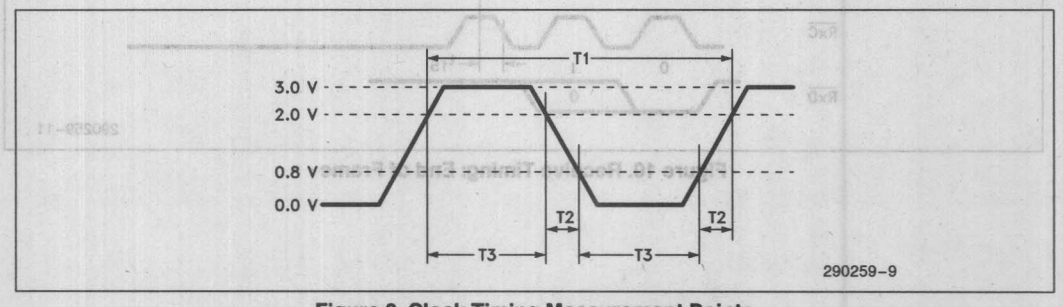


Figure 8. Clock Timing Measurement Points

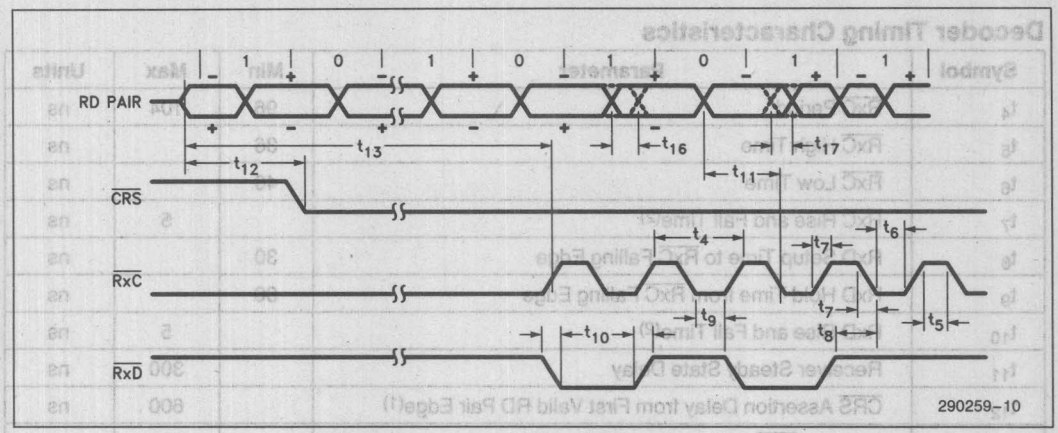


Figure 9. Receive Timing: Start of Frame

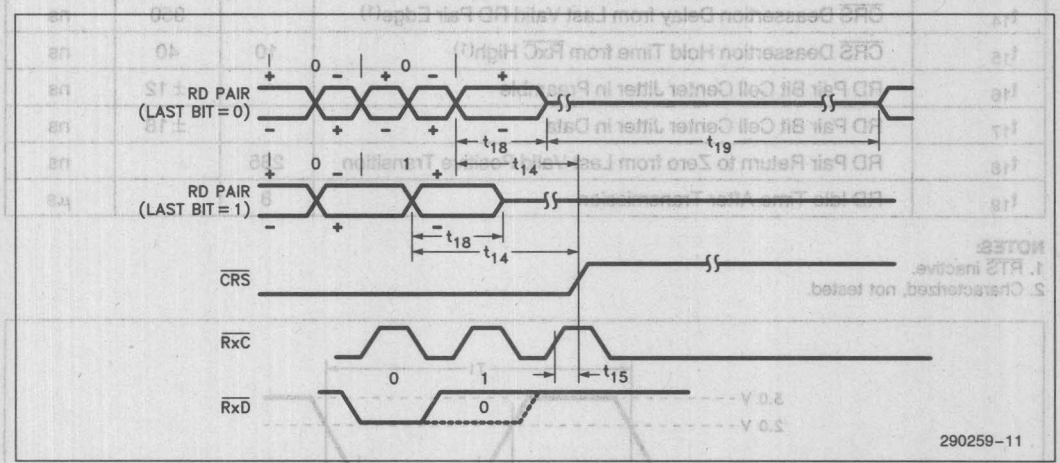


Figure 10. Receive Timing: End of Frame

Encoder Timing Characteristics

Symbol	Parameter	Min	Max	Units
t_{20}	$\overline{TxC}$ Period	99.99	100.01	ns
t_{21}	$\overline{TxC}$ Rise and Fall Time(2)		8	ns
t_{22}	$\overline{TxC}$ High Time	40		ns
t_{23}	$\overline{TxC}$ Low Time	40		ns
t_{24}	$\overline{RTS}$ Assertion to TD Pair Active	250	470	ns
t_{25}	Transmitter Steady State Delay	100	270	ns
t_{26}	TxD Setup Time to $\overline{TxC}$ Low	5		ns
t_{27}	TxD Hold Time from $\overline{TxC}$ Low	10		ns
t_{28}	TxD Setup Time from $\overline{TxC}$ High(1)	5		ns
t_{29}	TxD Hold Time from $\overline{TxC}$ High(1)	25		ns
t_{30}	$\overline{TxD}$ Rise and Fall Time(2)		10	ns
t_{31}	RTS Setup Time to $\overline{TxC}$ Low	5		ns
t_{32}	RTS Hold Time from $\overline{TxC}$ Low	10		ns
t_{33}	$\overline{RTS}$ Rise and Fall Time(2)		10	ns
t_{34}	TD Pair Output Jitter		±4	ns
t_{35}	$\overline{RTS}$ Deasserted to TD Pair Inactive	350	520	ns
t_{36}	TD Pair Held at Positive Differential at SOI	250	400	ns
t_{37}	TD Pair Return to ±50 mV after Transmission		4500	ns
t_{38}	$\overline{RTS}$ Deasserted to $\overline{CDT}$ Asserted (Heartbeat)	600	1600	ns
t_{39}	$\overline{CDT}$ Assertion Pulse Width (Heartbeat)	500	1500	ns

NOTES:
1. Manchester Encoder Disabled.
2. Characterized, not tested.

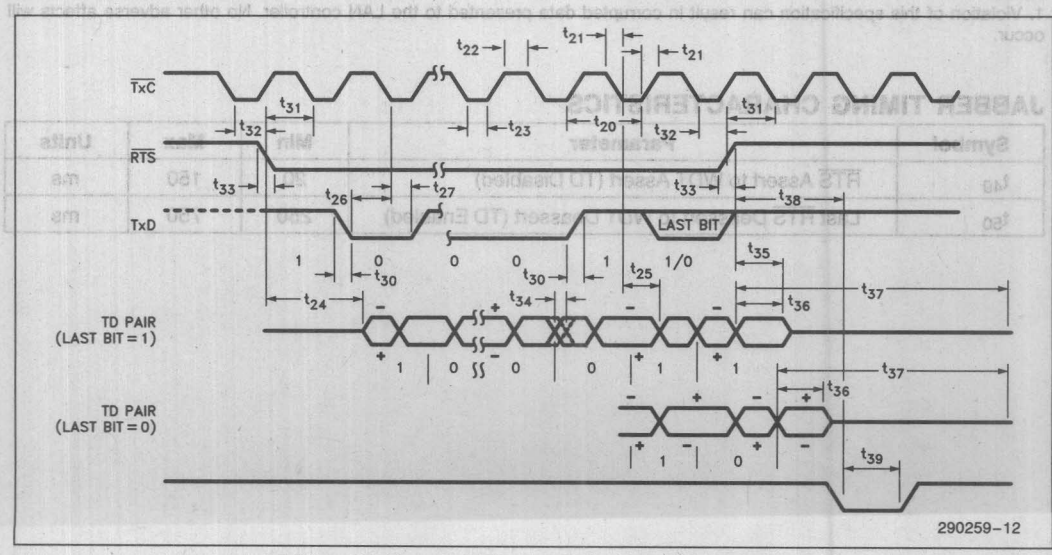


Figure 11. Transmit Timing: Manchester Encoding

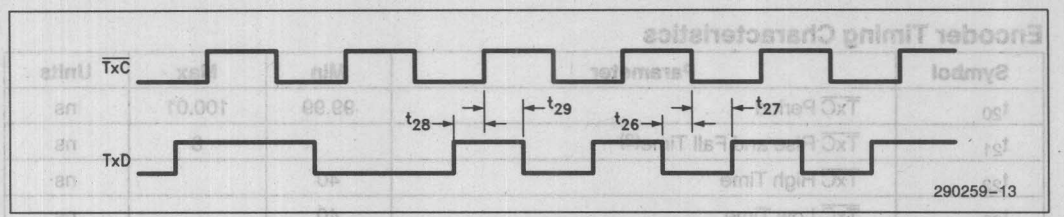


Figure 12. Transmit Timing: Non-Manchester Encoding

COLLISION TIMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
t_{41}	$\overline{CDT}$ Assertion from Onset of Collision	20	900	ns
t_{42}	$\overline{CDT}$ Deassertion from End of Collision	20	900	ns
t_{43}	$\overline{CRS}$ Deassertion from End of Collision(1)	20	900	ns

NOTE:
1. Both $\overline{RTS}$ and RD Pair Idle.

MODAL TIMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
t_{44}	RESET Pulse Width after V_{CC} Stable	1		ms
t_{45}	LID/WDTD, $\overline{MENC}$, LPBK2, $\overline{LPBK1}$ Setup to $\overline{RTS}$ Assert	10		µs
t_{46}	LID/WDTD, $\overline{MENC}$, LPBK2, $\overline{LPBK1}$ Hold from $\overline{RTS}$ Deassert	10		µs
t_{47}	LID/WDTD, $\overline{MENC}$, LPBK2, $\overline{LPBK1}$ Setup to RD Active(1)	10		µs
t_{48}	LID/WDTD, $\overline{MENC}$, LPBK2, $\overline{LPBK1}$ Hold from Rd Inactive(1)	10		µs

NOTE:
1. Violation of this specification can result in corrupted data presented to the LAN controller. No other adverse affects will occur.

JABBER TIMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
t_{49}	$\overline{RTS}$ Assert to WDT Assert (TD Disabled)	20	150	ms
t_{50}	Last $\overline{RTS}$ Deassert to WDT Deassert (TD Enabled)	250	750	ms

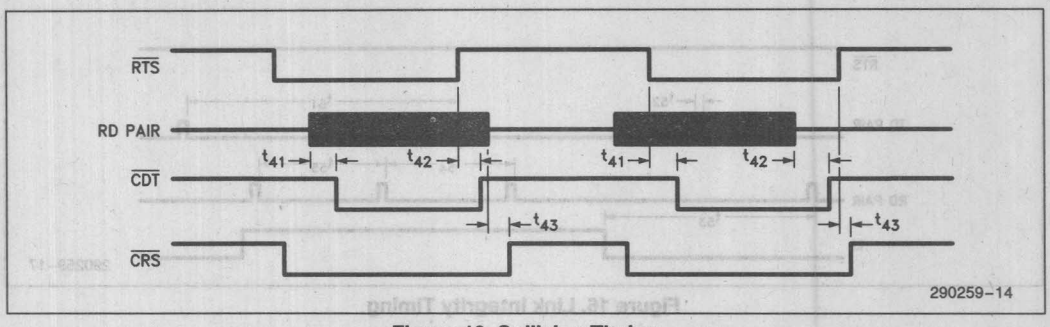


Figure 13. Collision Timing

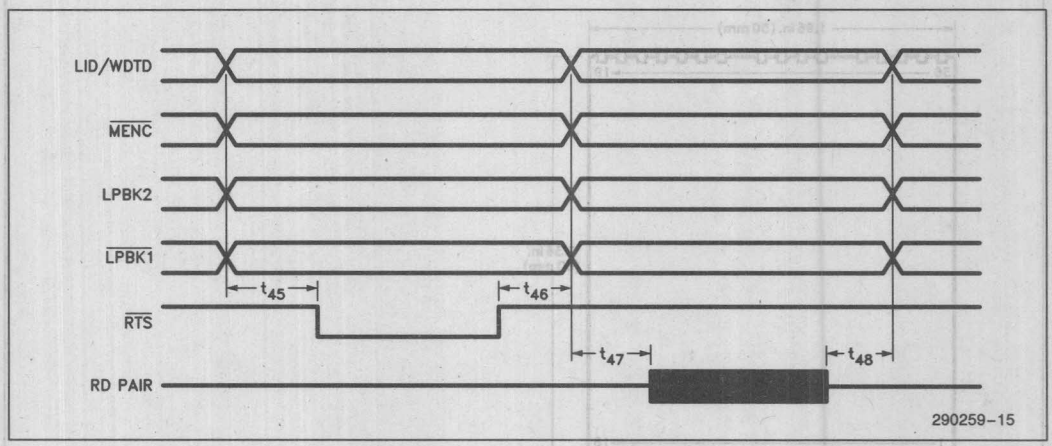


Figure 14. Modal Timing

LINK INTEGRITY TIMING CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
t_{51}	Last TD Edge to Link Test Pulse Transmission	8	24	ms
t_{52}	Link Test Pulse Width	80	120	ns
t_{53}	Last RD Edge to Link Fail (LI Assert)	50	150	ms
t_{54}	Minimum Idle Time between Consecutive Leakbeat Reception	6	8	ms
t_{55}	Maximum Idle Time between Consecutive Leakbeat Reception	24	150	ms

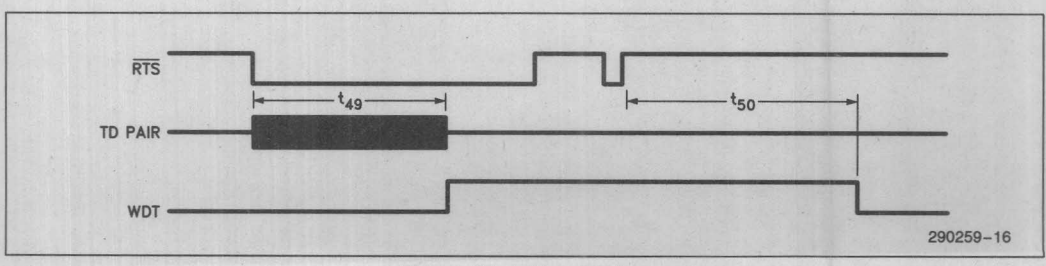


Figure 15. Jabber Timing

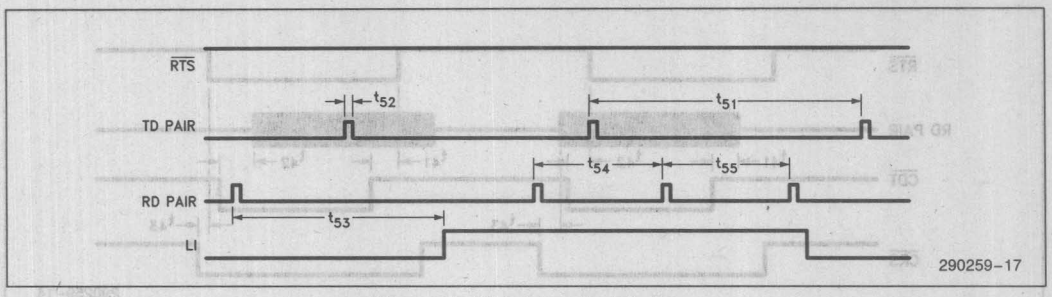


Figure 16. Link Integrity Timing

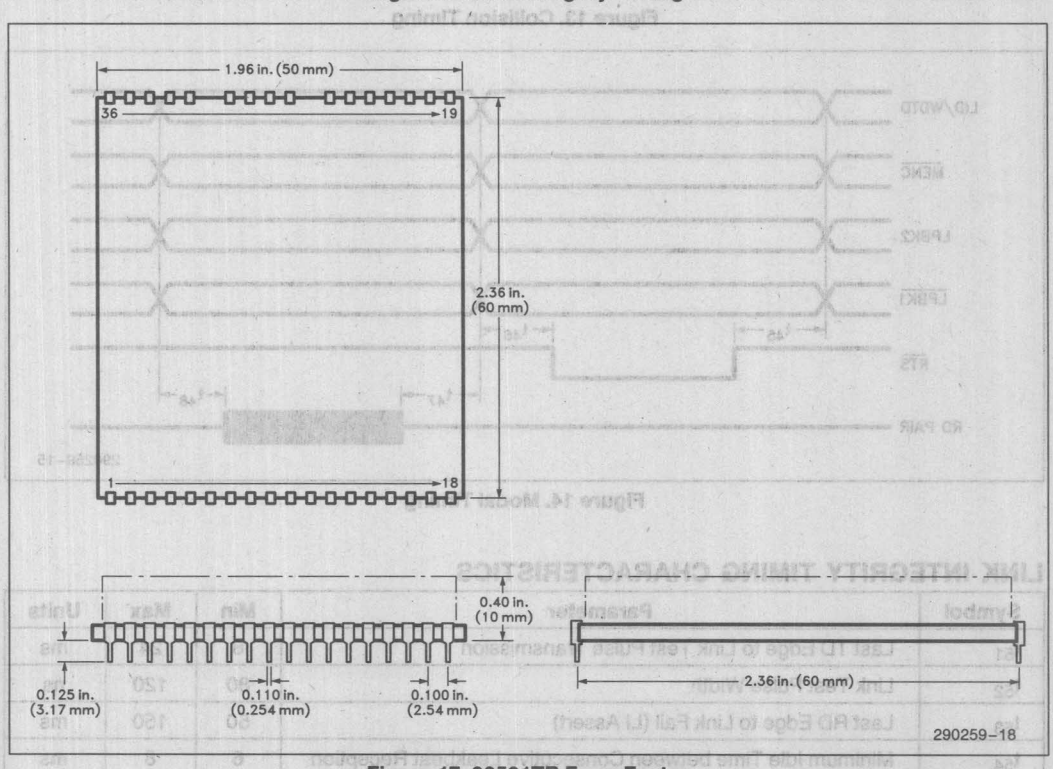


Figure 17. 82521TB Form Factor

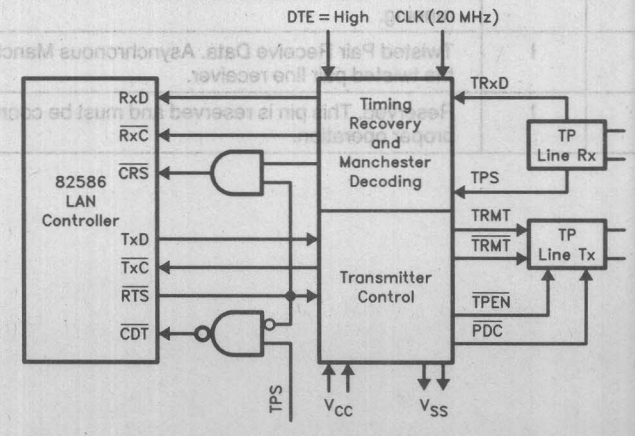
82504TA
Transceiver Serial Interface (TSI)

- **Designed for Twisted Pair Ethernet Applications**
  - **— Client Stations**
  - **— File Servers**
  - **— Bridges**
  - **— Twisted Pair Repeaters**
- **10 Mb/s Operation**
- **Interfaces Intel Ethernet LAN Controllers to Twisted Pair Link Segment**
- **Recovers Data and Clock Signal from Incoming Manchester Data**
- **Detects End-of-Packet Delimiter (IDL)**
- **Informs LAN Controller of Data Collisions and Loss of Signal**
- **Generates 10 MHz Transmit Clock**
- **Single 5V Supply, and Low-Power CMOS Processing**
- **Pin Compatible with AT&T T7210**

1

The Intel 82504TA Transceiver Serial Interface component (TSI) is intended for Twisted Pair Ethernet LAN applications using 10 Mb/s, Manchester coded data; for example, client stations, file servers, and repeaters. The 82504TA reduces design time by providing the serial interface functions required to connect the twisted pair interface circuitry to any of Intel's Ethernet LAN controllers, including the 82586, 82590, and 82592. It offers LAN system designers an easy way to upgrade existing Ethernet/Cheapernet products to take advantage of low-cost twisted pair wire. The TSI chip performs clock recovery and Manchester decoding of 10 Mb/s data, and produces NRZ data and clock signals for the LAN controller. The TSI also supports a predistortion method to prevent line overcharge, improving jitter performance. The 82504TA is pin compatible with the AT&T T7210. It is fabricated using low-power CMOS processing technology, and is available in 24-lead plastic DIP and 28-lead SOJ packages.

82504TA Block Diagram



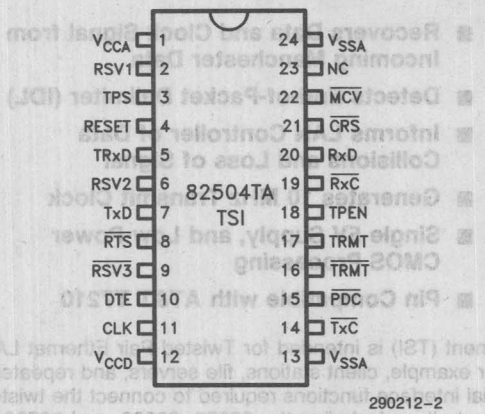
290212-1

Manufactured and tested for Intel by AT&T in accordance with AT&T internal standards.

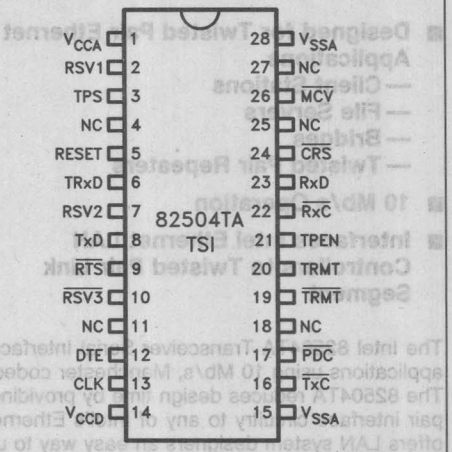
©Intel Corporation, 1989
©AT&T, 1989

November 1989
Order Number: 290212-001

24-Pin Plastic DIP and 28-Pin Plastic SOJ Pin Configurations



290212-2



290212-3

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
V_{CCA}	1		Analog V_{CC}. +5V power supply.
RSV1	2	I	Reserved. This pin is reserved and must be connected to V_{SSD} for proper operation.
TPS	3	I	Twisted Pair Sense. Active high. This pin is asserted when data is valid on TRxD (Twisted Pair Receive Data).
RESET	4	I	Reset. Active High. This pin is asserted to bring the TSI into a known state. It must be asserted for 1 ms while the clock is running.
TRxD	5	I	Twisted Pair Receive Data. Asynchronous Manchester data from the twisted pair line receiver.
RSV2	6	I	Reserved. This pin is reserved and must be connected to V_{SSD} for proper operation.

NOTES:
I = Input
O = Output

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
TxD	7	I	Transmit Data. Manchester encoded data from the 82586 (or other Ethernet controller). This pin is directly connected to the TxD controller output.
$\overline{RTS}$	8	I	Request to Send. Active low. This signal is synchronous to $\overline{TxC}$, and enables data transmission on the twisted pair link segment.
$\overline{RSV3}$	9	I	Reserved. This pin is reserved and must be connected to V_{CCD} for proper operation.
DTE	10	I	Data Terminal Equipment. This pin should be connected to V_{CCD} if the TSI is used in a DTE, or to V_{SSD} if used in a repeater.
CLK	11	I	Clock. A 20 MHz ±0.01% input clock used for precision timing and encoded data transmission.
V_{CCD}	12		Digital V_{CC}. +5V Power Supply.
V_{SSD}	13		Digital Ground.
$\overline{TxC}$	14	O	Transmit Clock. A 10 MHz clock output tied directly to the $\overline{TxC}$ pin of the Intel Ethernet LAN Controller.
$\overline{PDC}$	15	O	Predistortion Control. This signal is used to reduce jitter in a twisted pair environment by preventing line overcharge. This pin is asserted for the first 50 ns of any pulse on the TRMT pair. This allows the TP line drivers to reduce their output voltage during the last 50 ns of 100 ns Manchester pulses. $\overline{PDC}$ will not produce glitches during consecutive 50 ns pulses.
$\overline{TRMT}$ TRMT	16 17	O O	Twisted Pair Transmit Pair. Serial Manchester encoded data generated for the twisted pair line drivers.
$\overline{TPEN}$	18	O	Twisted Pair Enable. Active low. This pin enables the line drivers.
$\overline{RxC}$	19	O	Receive Clock. A 10 MHz clock connected directly to the $\overline{RxC}$ input of the Intel Ethernet LAN Controller. This clock is the recovered clock from TRxD.
RxD	20	O	Receive Data. NRZ data passed to the Intel Ethernet LAN Controller. This pin is directly connected to the RxD pin on the controller.
$\overline{CRS}$	21	O	Carrier Sense. Active low. A signal that alerts the Intel Ethernet LAN Controller that the twisted pair link is active. This pin is directly connected to the $\overline{CRS}$ input of the controller.
$\overline{MCV}$	22	O	Manchester Code Violation. Active low. This signal indicates the presence of Manchester code violations.
NC	23		Not Connected.
V_{SSA}	24		Analog Ground.

The following are two possible vendors:

- Fox Electronics
  5842 Corporation Circle
  Fort Myers, FL 33905
- M-Tron Industries, Inc.
  Yankton, SD 57078

An external TTL-compatible 20 MHz ±0.01% clock with a duty cycle of 40/60 or better can also be used.

Transmit Section

MANCHESTER RETIMING AND PREDISTORTION

The transmit section of the 82504TA is controlled by the $\overline{RTS}$ signal generated by the Ethernet LAN controller (Intel's 82586, 82592, etc). When $\overline{RTS}$ is asserted, the 82504TA begins clocking in Manchester data from the controller on the TxD input. The TxD signal is sampled on every transition of the 10 MHz $\overline{TxC}$ signal. The serial data is then retimed by the 20 MHz input clock, and sent to the line drivers via the TRMT and $\overline{TRMT}$ pins. The enable signal for the line drivers, $\overline{TPEN}$, asserts two bit times after the assertion of $\overline{RTS}$ to allow the input Manchester data to settle. At the end of the packet, $\overline{TPEN}$ remains asserted for three bit times to make allowance for device latency, and to append the end of packet symbol (IDL) to the data packet.

Another signal, Predistortion Control ($\overline{PDC}$), is also generated by the transmit section. Predistortion is a technique for reducing jitter by preventing line overcharging during "fat" (100 ns) Manchester pulses. $\overline{PDC}$ is asserted during the first 50 ns of any pulse on the TRMT outputs; i.e., it is asserted throughout "thin" (50 ns) pulses and during the first half of "fat" pulses. This permits the twisted pair line driver to reduce its output drive during the second half of "fat" pulses. Internal circuitry prevents glitches on $\overline{PDC}$.

Receive Section

MANCHESTER DECODING AND CLOCK RECOVERY

The Receive section of the 82504TA is enabled when incoming data from the twisted pair asserts the Twisted Pair Sense (TPS) signal. Manchester data decoding and clock recovery begin on the serial data from the Twisted Pair Receive Data (TRxD) input. $\overline{RxC}$ changes from its free running state to its locked state during the first two bit-times. $\overline{CRS}$ goes active after two bit times to guarantee reception of valid data after $\overline{RxC}$ clock stabilization. The decoded NRZ data is sent to the LAN controller on the RxD line along with the recovered clock signal, $\overline{RxC}$.

The end of packet is detected by the presence of the IDL symbol or by the deassertion of TPS. After three bit times $\overline{CRS}$ will be deasserted synchronously with $\overline{RxC}$, then $\overline{RxC}$ returns to its free running state.

To interface with a LAN controller that expects $\overline{CRS}$ to be asserted in response to its own transmission—Intel controllers are software configurable either way—the $\overline{CRS}$ signal from the TSI should be AND'd with the $\overline{RTS}$ signal from the controller (as shown in Figure 2); this way, $\overline{CRS}$ to the controller will assert during both transmission or reception. This is the normal mode of operation for coaxial Ethernet environments.

APPLICATION EXAMPLE

A typical DTE receiver design is shown in Figure 2. The incoming signal from the twisted pair wire passes through a common-mode choke and an isolation transformer for noise reduction. This signal runs through another filter. The filter output runs directly to a line receiver to establish a data channel, and through a DC offset to another line receiver for a squelch channel. The squelch channel is used for noise rejection, and detecting valid incoming

data. The line receivers on both the data channel and squelch channel convert the differential signal to TTL-compatible signals.

When the incoming signal level is above the comparator's preset threshold, the comparator output triggers a Retriggerable Monostable Multivibrator. The multivibrator then asserts the TPS signal for the 82504TA. The TPS signal remains asserted for two bit times past the last input transition.

The TPS signal is used to gate the data-channel line driver from the TRxD signal to ensure proper operation. Further, the V_{IH} level of TRxD should be held between 1.8V and 2.4V. In the example shown in Figure 2 this is accomplished by using a 100Ω pull-down resistor on the output of the AND gate (74F08), which is used to gate the data channel with TPS.

Collision Detect

Collision detection in the twisted pair environment occurs from simultaneous transmission and reception on the twisted pair wires. This is indicated by the assertion of both TPS and $\overline{RTS}$. The simple logic circuit shown in Figure 2 can detect such collisions.

Interface Example

Figure 3 shows a typical DTE implementation circuit. When designing this type of circuit, considerable attention must be paid to power supply noise reduction, capacitive decoupling, and the layout of the line driver/receiver to the interface connector (RJ-45).

REPEATER MODE

Operation in Repeater Mode

The 82504TA can be used when the DTE pin is not asserted. There are two principal differences in this mode of operation. First, the deassertion of $\overline{CRS}$ is not synchronized to $\overline{RxC}$—this, on the average, allows $\overline{CRS}$ to deassert one bit time earlier. Second, $\overline{TPEN}$ assertion occurs two bit times earlier than in DTE mode.

Application Example

A repeater design using the 82505TA Multiport Repeater controller (MPR) requires the services of an 82504 TSI. The TSI provides the Manchester decoder and clock recovery for the MPR. Figure 4 shows the appropriate interface circuitry.

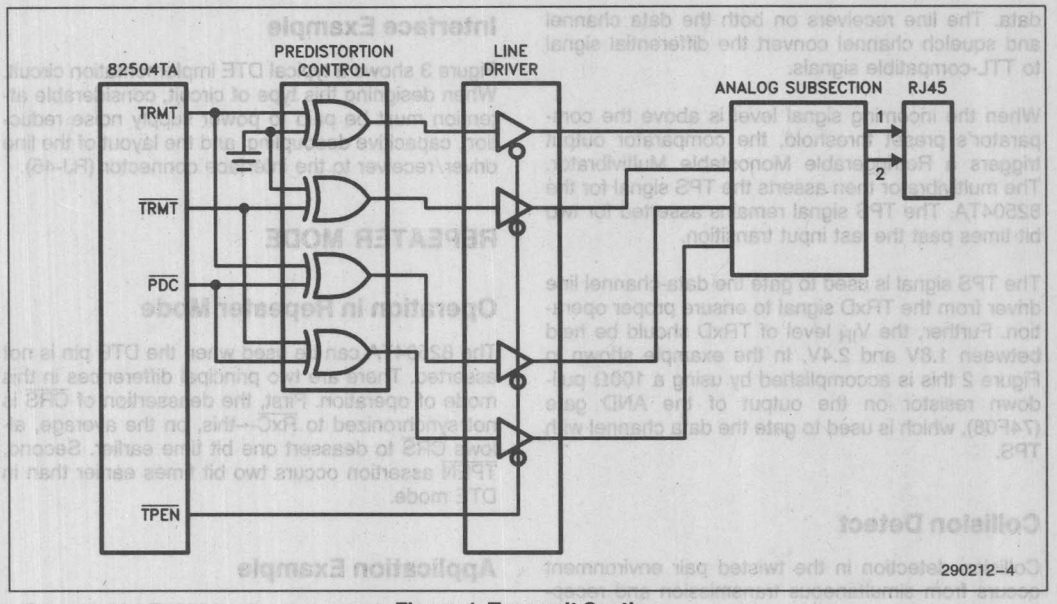


Figure 1. Transmit Section

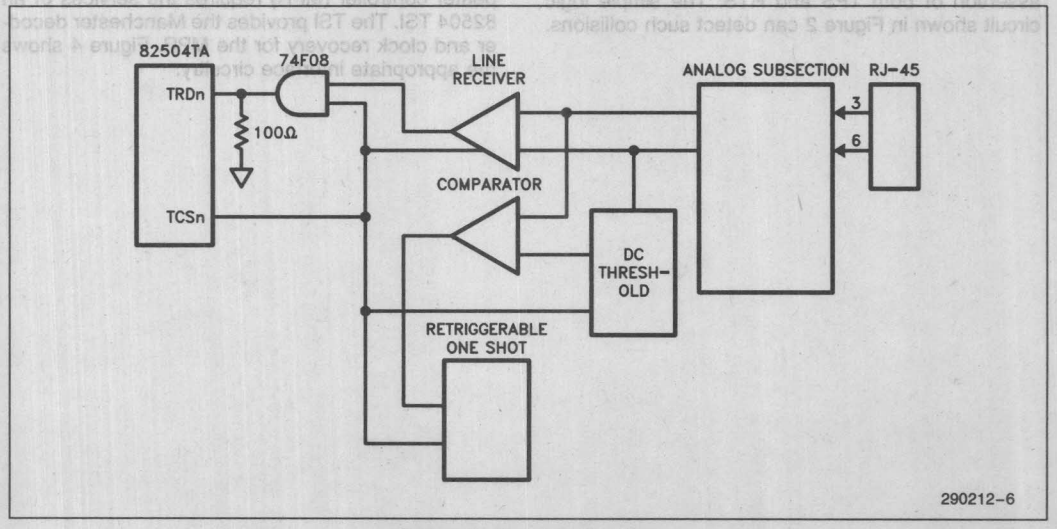


Figure 2. Receive Section

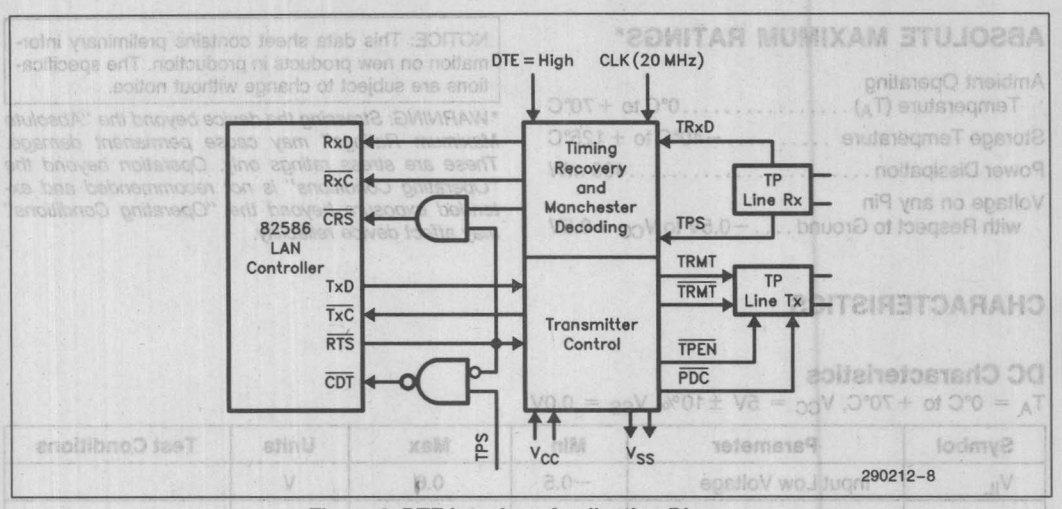


Figure 3. DTE Interface Application Diagram

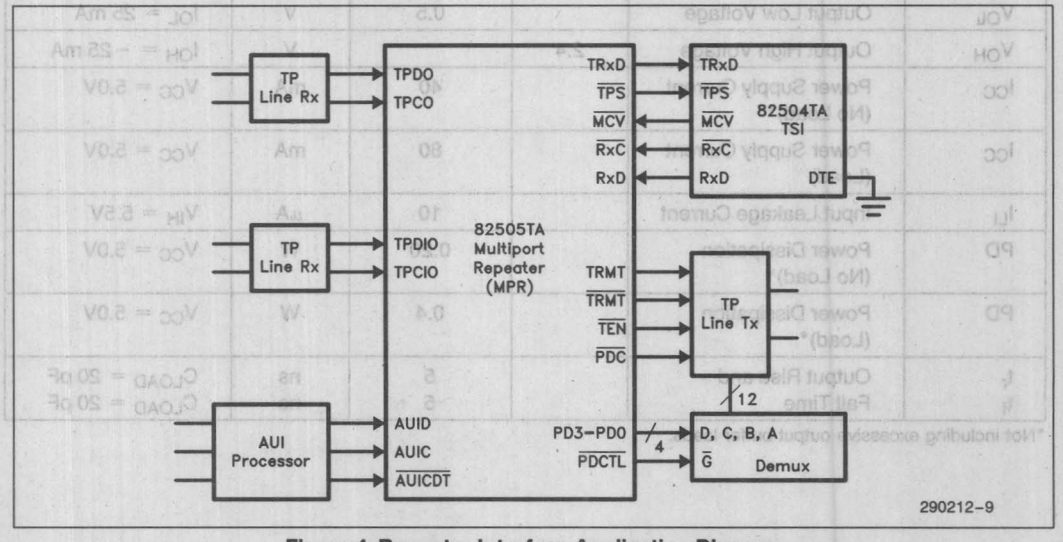


Figure 4. Repeater Interface Application Diagram

ABSOLUTE MAXIMUM RATINGS*

Ambient Operating Temperature (T_A) . . . . . 0°C to +70°C

Storage Temperature . . . . . −40°C to +125°C

Power Dissipation . . . . . 400 mW

Voltage on any Pin with Respect to Ground . . . . −0.5V to V_{CC} +0.5V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

CHARACTERISTICS

DC Characteristics

T_A = 0°C to +70°C, V_{CC} = 5V ±10%, V_{SS} = 0.0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.6	V	
V_{IH}	Input High Voltage	2.0	V_{DD} +0.5	V	
V_{OL}	Output Low Voltage		0.5	V	I_{OL} = 25 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −25 mA
I_{CC}	Power Supply Current (No Load)*		40	mA	V_{CC} = 5.0V
I_{CC}	Power Supply Current (Load)		80	mA	V_{CC} = 5.0V
I_{LI}	Input Leakage Current		10	μA	V_{IH} = 5.5V
PD	Power Dissipation (No Load)*		0.20	W	V_{CC} = 5.0V
PD	Power Dissipation (Load)*		0.4	W	V_{CC} = 5.0V
t_r t_f	Output Rise and Fall Time		5 5	ns ns	C_{LOAD} = 20 pF C_{LOAD} = 20 pF

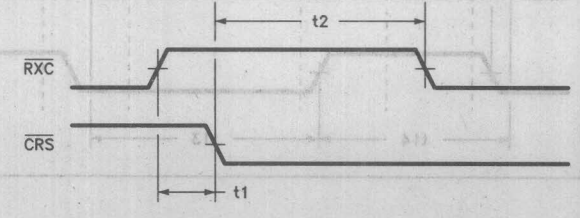
*Not including excessive output buffer loads.

DECODER TIMING CHARACTERISTICS

(Measurements are from 50% points, unless otherwise noted.)

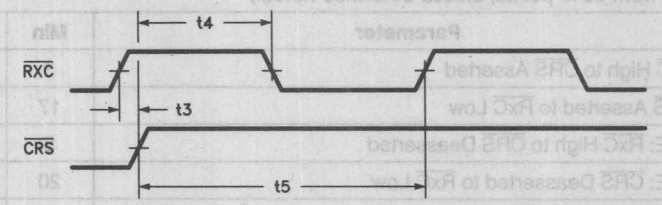
Symbol	Parameter	Min	Max	Units
t1	$\overline{RxC}$ High to $\overline{CRS}$ Asserted	3	19	ns
t2	$\overline{CRS}$ Asserted to $\overline{RxC}$ Low	17	40	ns
t3	DTE: $\overline{RxC}$ High to $\overline{CRS}$ Deasserted	3	19	ns
t4	DTE: $\overline{CRS}$ Deasserted to $\overline{RxC}$ Low	20	65	ns
t5	DTE: $\overline{CRS}$ Deasserted to $\overline{RxC}$ High	195	345	ns
t6	$\overline{RxC}$ High Pulse Width as Captured Data Clock	36	45	ns
t7	$\overline{RxC}$ Low Pulse Width as Captured Data Clock	38	80	ns
t8	$\overline{RxC}$ Period as Captured Data Clock	78	124	ns
t9	$\overline{RxC}$ High Pulse Width as Free Oscillating Clock	43	73	ns
t10	$\overline{RxC}$ Low Pulse Width as Free Oscillating Clock	172	276	ns
t11	$\overline{RxC}$ Period as Free Oscillating Clock	215	349	ns
t12	$\overline{RxD}$ Transition to $\overline{RxC}$ High	−5	5	ns
t13	$\overline{RxC}$ Low to $\overline{RxD}$ Transition	30	85	ns
t14	$\overline{RxD}$ Transition to $\overline{RxC}$ Low	30	50	ns
t15	TRxD Midbit Transition to $\overline{RxC}$ Low	86	130	ns
t16	TPS Asserted to TRxD Sampled	−5	20	ns
t17	TRxD Preamble Transition to $\overline{CRS}$ Asserted	53	95	ns
t18	DTE: Beginning of IDL to $\overline{CRS}$ Deasserted (Last Bit = 0) DTE: Beginning of IDL to $\overline{CRS}$ Deasserted (Last Bit = 1)	230 280	320 390	ns ns
t19	Repeater: Beginning of IDL to $\overline{CRS}$ Deasserted (Last Bit = 0) Repeater: Beginning of IDL to $\overline{CRS}$ Deasserted (Last Bit = 1)	180 170	220 220	ns ns
t20	Midbit to Midbit Transition on TRxD	80	120	ns
t21	Boundary to Midbit Transition on TRxD	30	70	ns

Negative-Transition $\overline{CRS}$ Timing Relative to $\overline{RxC}$



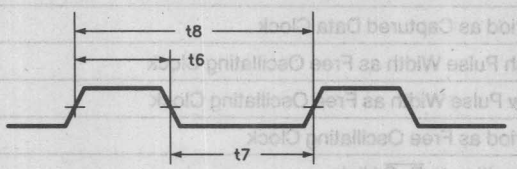
290212-10

DTE Positive-Transition $\overline{CRS}$ Timing Relative to $\overline{RxC}$



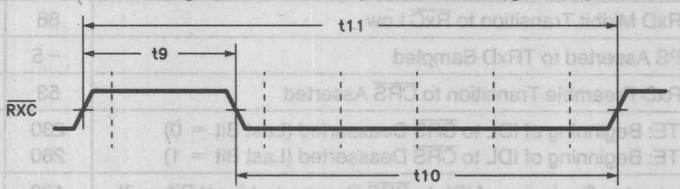
290212-11

$\overline{RxC}$ Timing Measurements (Captured Data Clock)



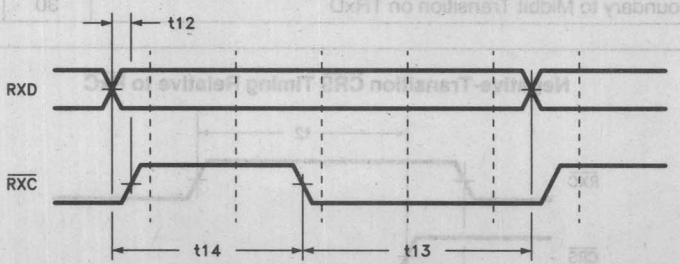
290212-12

$\overline{RxC}$ Timing Measurements (Free Oscillating Clock)



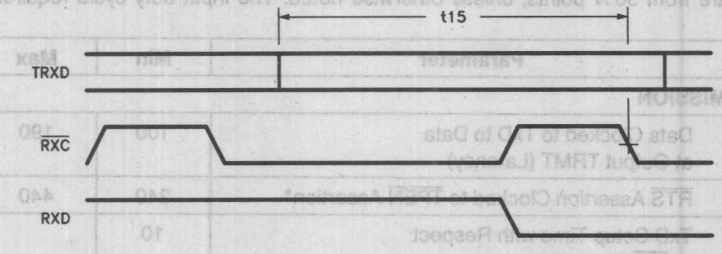
290212-13

RxD Timing Relative to $\overline{RxC}$



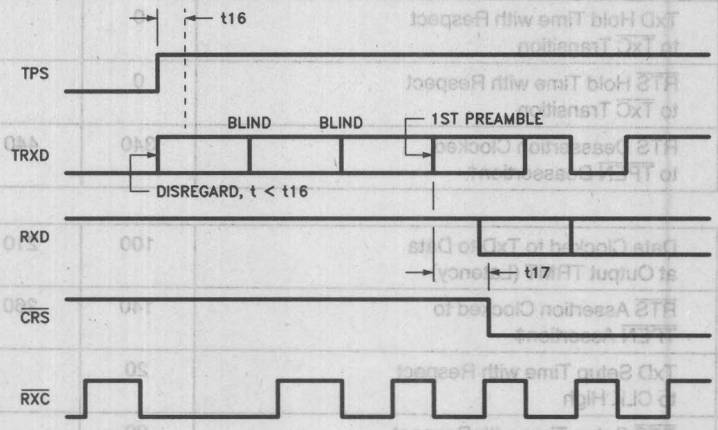
290212-14

Latency-Definition Timing Measurements



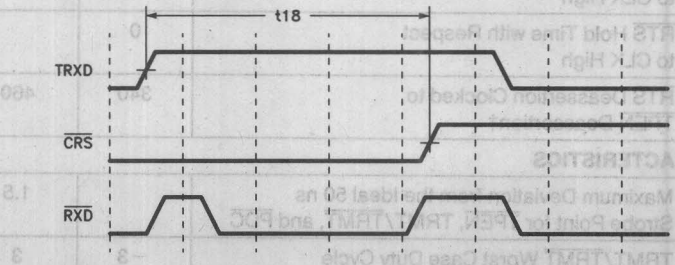
290212-15

Start-of-Packet Timing Measurements



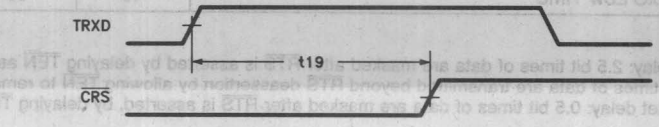
290212-16

DTE End-of-Normal-Packet Timing Measurements



290212-17

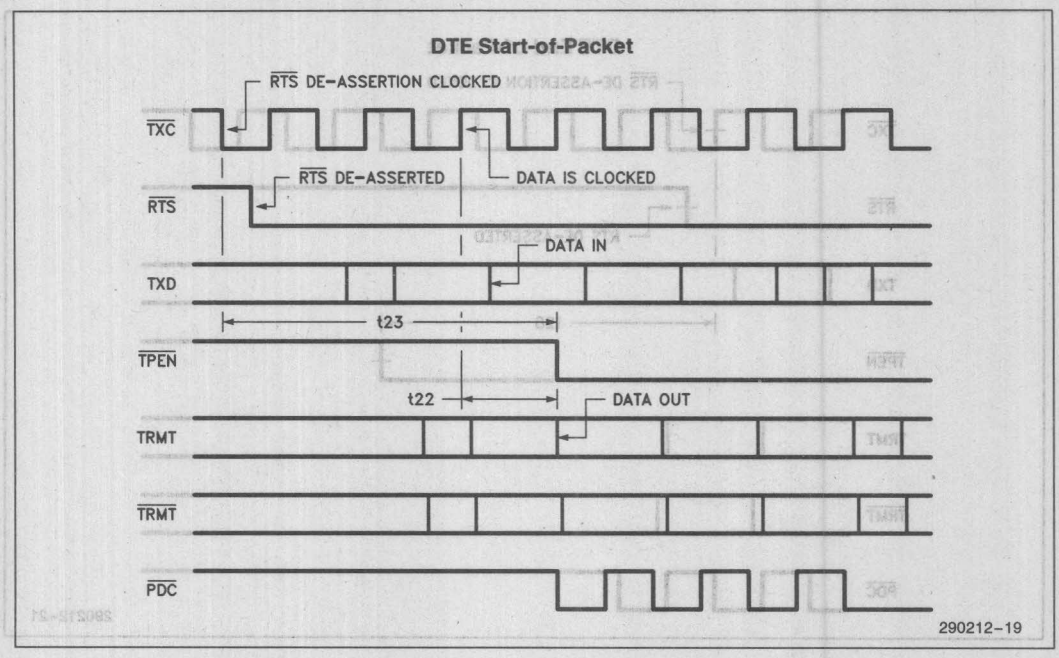
Repeater CRS Timing Relative to IDL

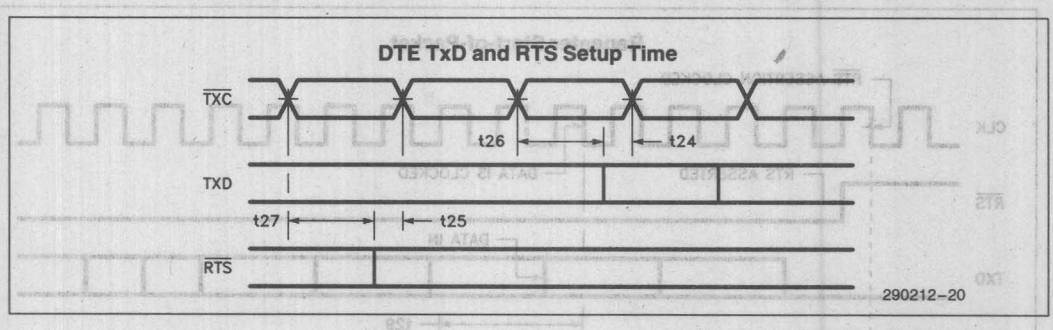


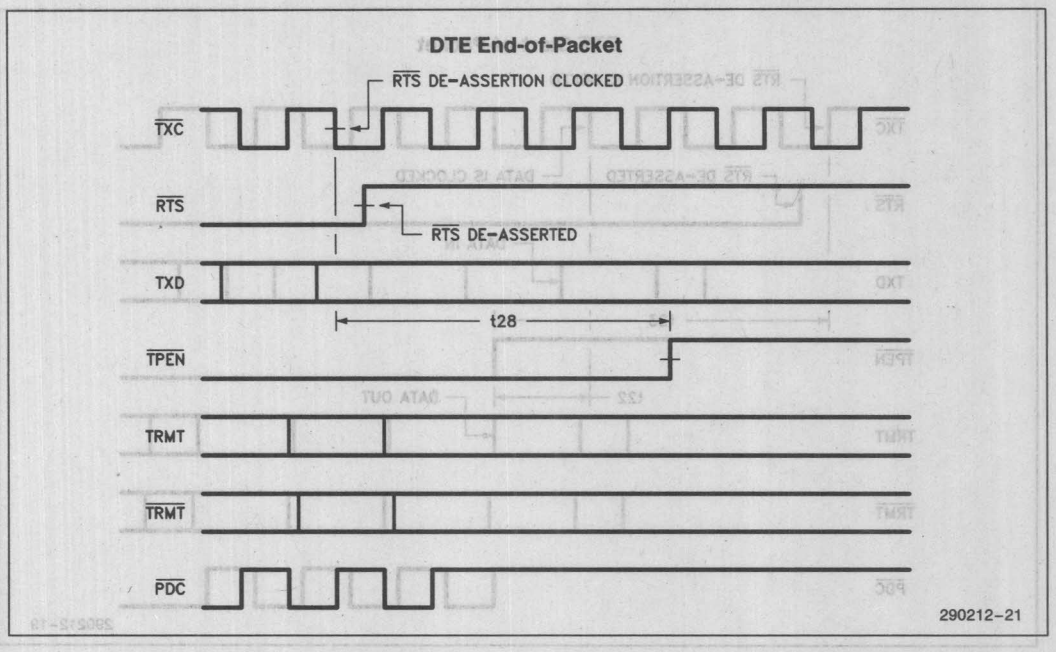
290212-18

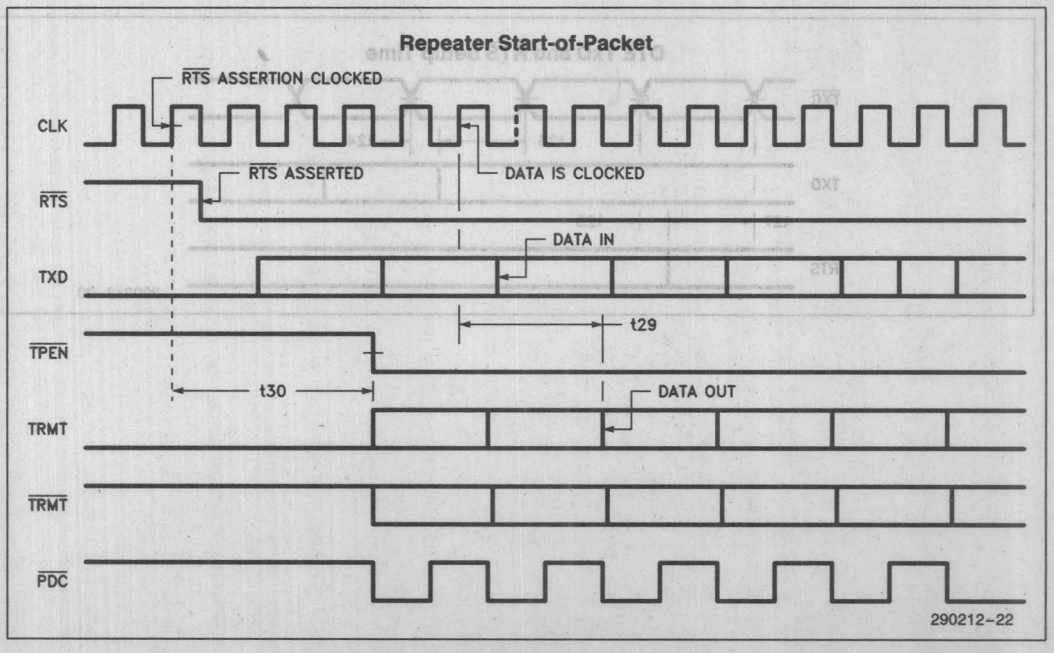
	to $\overline{TxC}$ Transition			
t26	TxD Hold Time with Respect to $\overline{TxC}$ Transition	0		ns
t27	$\overline{RTS}$ Hold Time with Respect to $\overline{TxC}$ Transition	0		ns
t28	$\overline{RTS}$ Deassertion Clocked to $\overline{TPEN}$ Deassertion†	340	440	ns
REPEATER				
t29	Data Clocked to TxD to Data at Output TRMT (Latency)	100	210	ns
t30	$\overline{RTS}$ Assertion Clocked to $\overline{TPEN}$ Assertion‡	140	260	ns
t31	TxD Setup Time with Respect to CLK High	20		ns
t32	$\overline{RTS}$ Setup Time with Respect to CLK High	20		ns
t33	TxD Hold Time with Respect to CLK High	0		ns
t34	$\overline{RTS}$ Hold Time with Respect to CLK High	0		ns
t35	$\overline{RTS}$ Deassertion Clocked to $\overline{TPEN}$ Deassertion†	340	460	ns
OUTPUT CHARACTERISTICS				
t36	Maximum Deviation from the Ideal 50 ns Strobe Point for $\overline{TPEN}$, TRMT/$\overline{TRMT}$, and $\overline{PDC}$		1.5	ns
t37–t38	TRMT/$\overline{TRMT}$ Worst Case Duty Cycle Mismatch, 10 pF Load	−3	3	ns
t39	$\overline{TxC}$ High Time	40	60	ns
t40	$\overline{TxC}$ Low Time	40	60	ns

NOTES:
* DTE start-of-packet delay: 2.5 bit times of data are masked after $\overline{RTS}$ is asserted by delaying $\overline{TEN}$ assertion.
† End of Packet: 2.5 bit times of data are transmitted beyond $\overline{RTS}$ deassertion by allowing $\overline{TEN}$ to remain asserted.
‡ Repeater start-of-packet delay: 0.5 bit times of data are masked after $\overline{RTS}$ is asserted, by delaying $\overline{TEN}$ assertion.

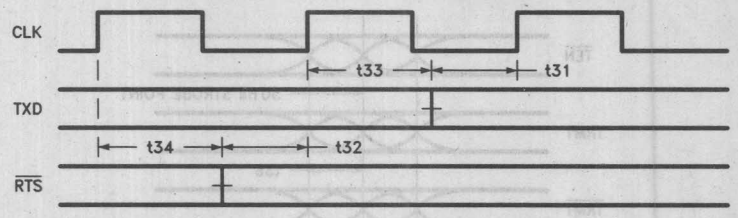

DTE Start-of-Packet
RTS DE-ASSERTION CLOCKED
TXC
RTS DE-ASSERTED
DATA IS CLOCKED
RTS
DATA IN
TXD
t23
TPEN
t22
DATA OUT
TRMT
TRMT
PDC
290212-19


DTE TxD and RTS Setup Time
TXC
t26
t24
TXD
t27
t25
RTS
290212-20


DTE End-of-Packet
RTS DE-ASSERTION CLOCKED
TXC
RTS
RTS DE-ASSERTED
TXD
t28
TPEN
TRMT
TRMT
PDC
290212-21

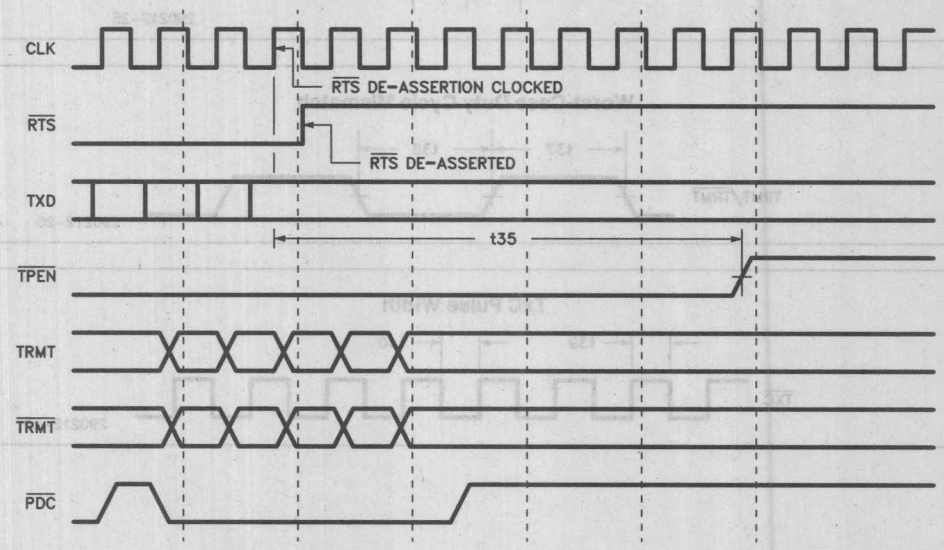

Repeater Start-of-Packet
RTS ASSERTION CLOCKED
CLK
RTS ASSERTED
DATA IS CLOCKED
RTS
DATA IN
TXD
t29
TPEN
t30
DATA OUT
TRMT
TRMT
PDC
290212-22

Repeater TxD and $\overline{RTS}$ Setup Time

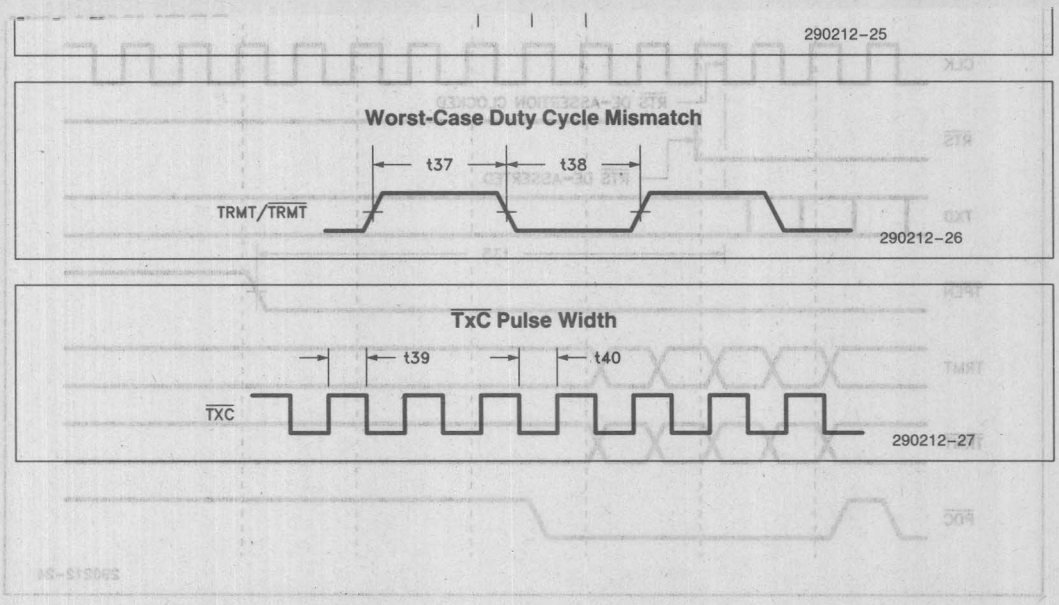


290212-23

Repeater End-of-Packet



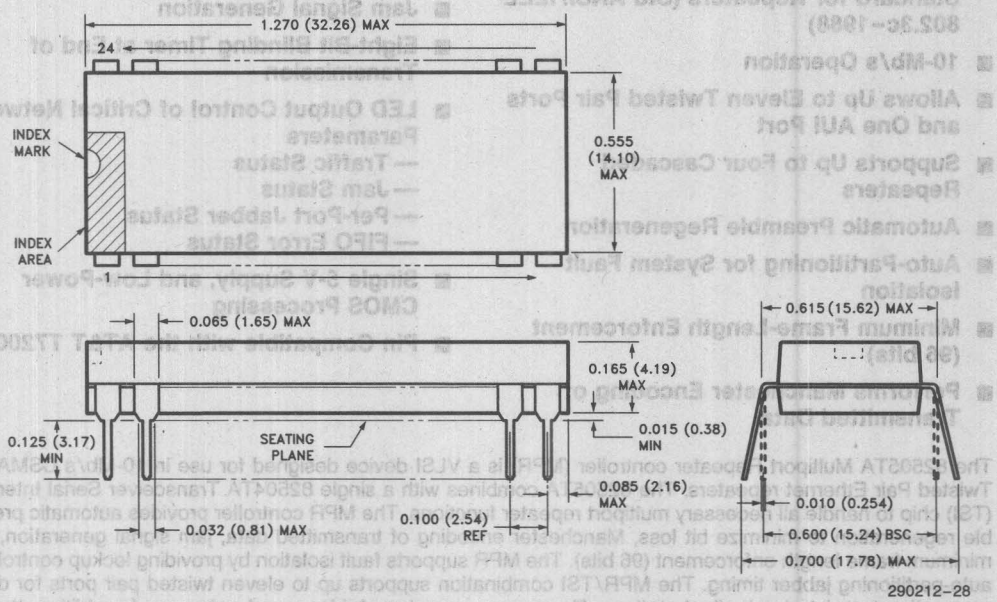
290212-24


290212-25
Worst-Case Duty Cycle Mismatch
t37
t38
TRMT/TRMT
290212-26
TxC Pulse Width
t39
t40
TXC
290212-27

OUTLINE DIAGRAM

24-Pin Plastic DIP

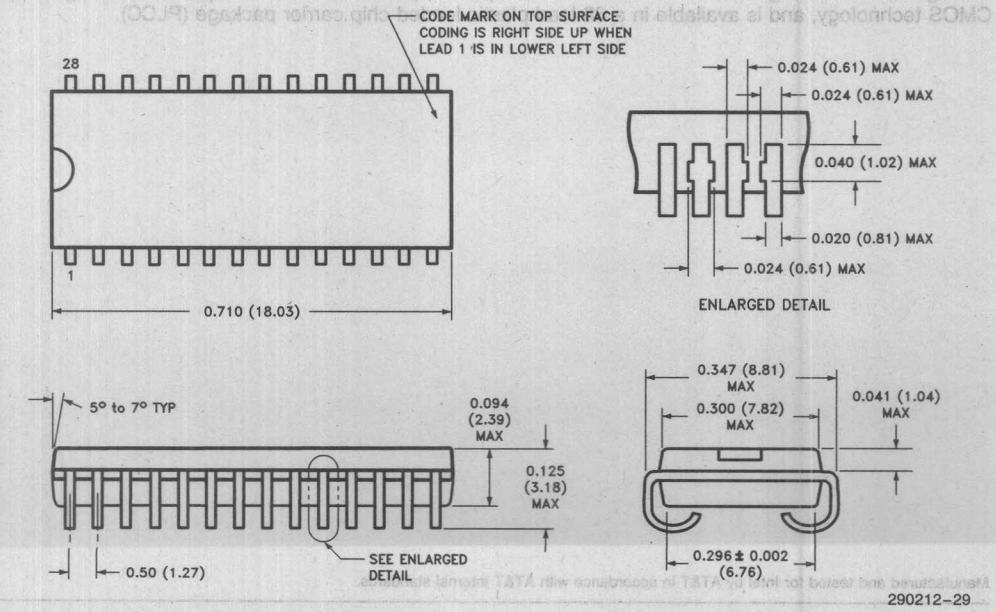
Dimensions are in inches and (millimeters)



290212-28

28-Pin Plastic SOJ

Dimensions are in inches and (millimeters)

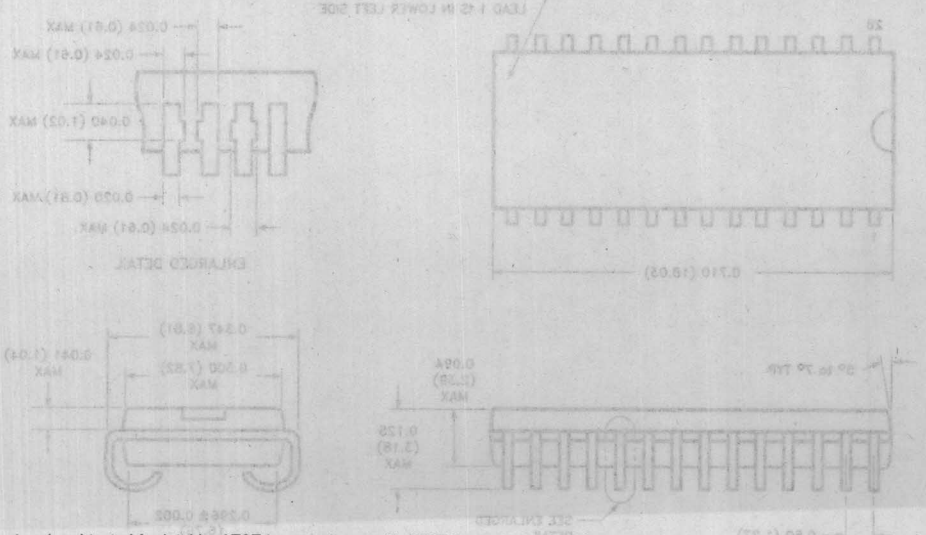


290212-29

82505TA
MULTIPORT REPEATER CONTROLLER (MPR)

- Complies with IEEE 802.3 CSMA/CD Standard for Repeaters (Std ANSI/IEEE 802.3c–1988)
- 10-Mb/s Operation
- Allows Up to Eleven Twisted Pair Ports and One AUI Port
- Supports Up to Four Cascaded Repeaters
- Automatic Preamble Regeneration
- Auto-Partitioning for System Fault Isolation
- Minimum Frame-Length Enforcement (96 bits)
- Performs Manchester Encoding of Transmitted Data
- Pin-Selectable FIFO Fill Level
- Jam Signal Generation
- Eight-Bit Blinding Timer at End of Transmission
- LED Output Control of Critical Network Parameters
  - — Traffic Status
  - — Jam Status
  - — Per-Port Jabber Status
  - — FIFO Error Status
- Single 5-V Supply, and Low-Power CMOS Processing
- Pin Compatible with the AT&T T7200

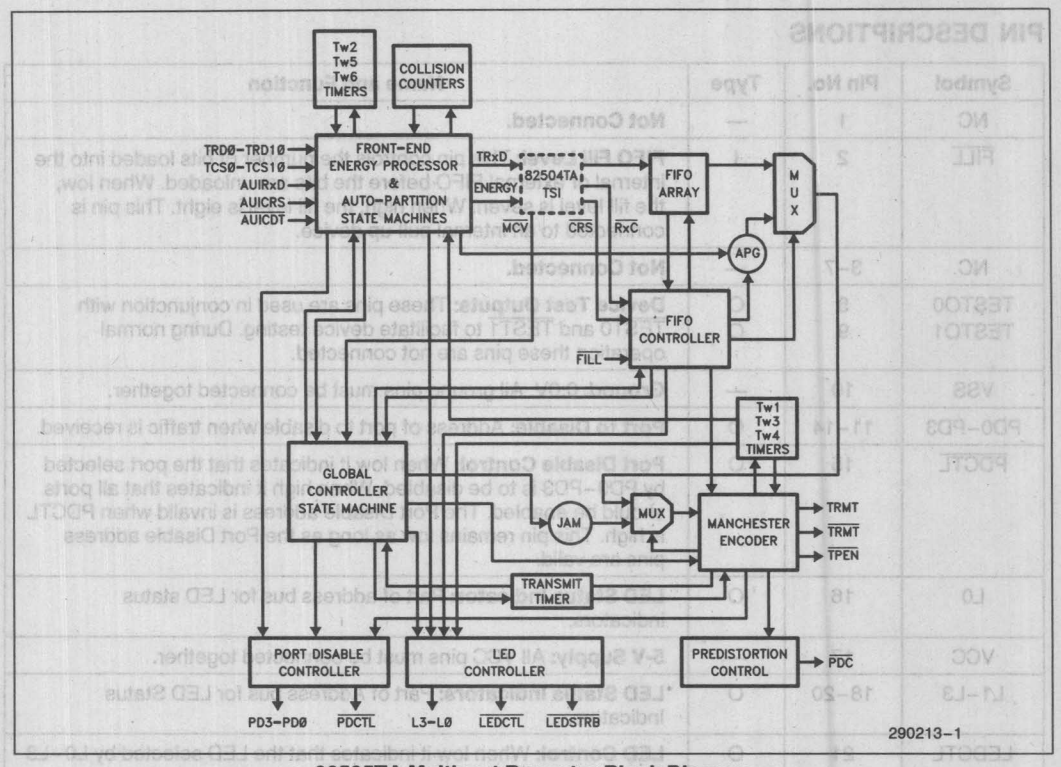
The 82505TA Multiport Repeater controller (MPR) is a VLSI device designed for use in 10-Mb/s CSMA/CD Twisted Pair Ethernet repeaters. The 82505TA combines with a single 82504TA Transceiver Serial Interface (TSI) chip to handle all necessary multiport repeater functions. The MPR controller provides automatic preamble regeneration to minimize bit loss, Manchester encoding of transmitted data, jam signal generation, and minimum frame length enforcement (96 bits). The MPR supports fault isolation by providing lockup control and auto-partitioning jabber timing. The MPR/TSI combination supports up to eleven twisted pair ports for direct connection to twisted pair client stations, file servers, repeaters, bridges, and gateways. In addition, the set supports one AUI port for interfacing twisted pair networks to existing Ethernet (IEEE 802.3 TYPE 10BASE5) or Cheapernet (IEEE 802.3 TYPE 10BASE2) networks. The MPR offers pin selectable FIFO fill levels and LED output control of traffic status, jam status, per-port jabber status, and FIFO error status for simplified network management and diagnostics. The 82505TA Multiport Repeater controller is fabricated using low-power CMOS technology, and is available in a 68-lead plastic leaded chip carrier package (PLCC).



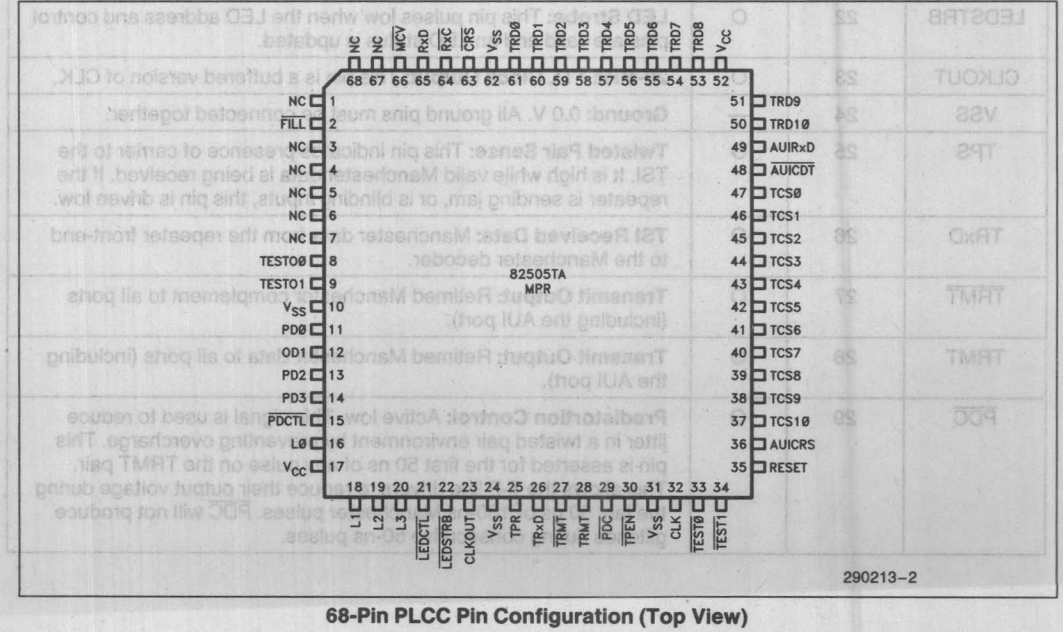
Manufactured and tested for Intel by AT&T in accordance with AT&T internal standards.

©Intel Corporation, 1989
©AT&T, 1989

April 1989
Order Number: 290213-001



82505TA Multiport Repeater Block Diagram



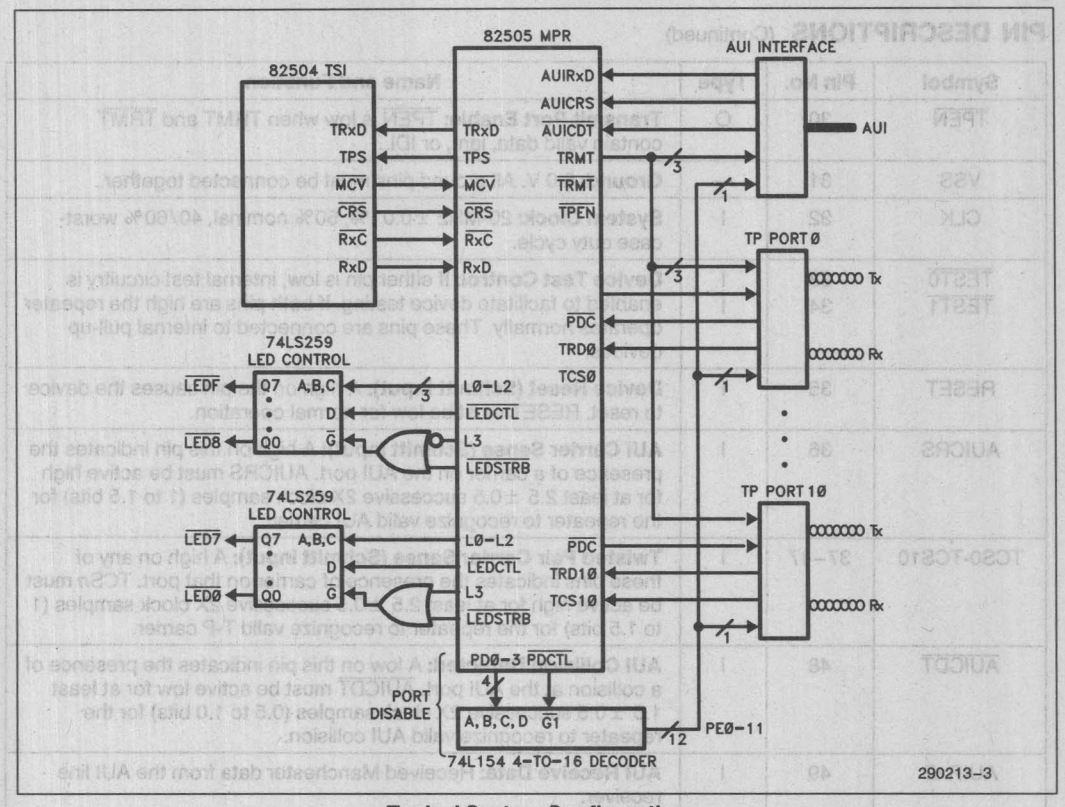
68-Pin PLCC Pin Configuration (Top View)

PIN DESCRIPTIONS

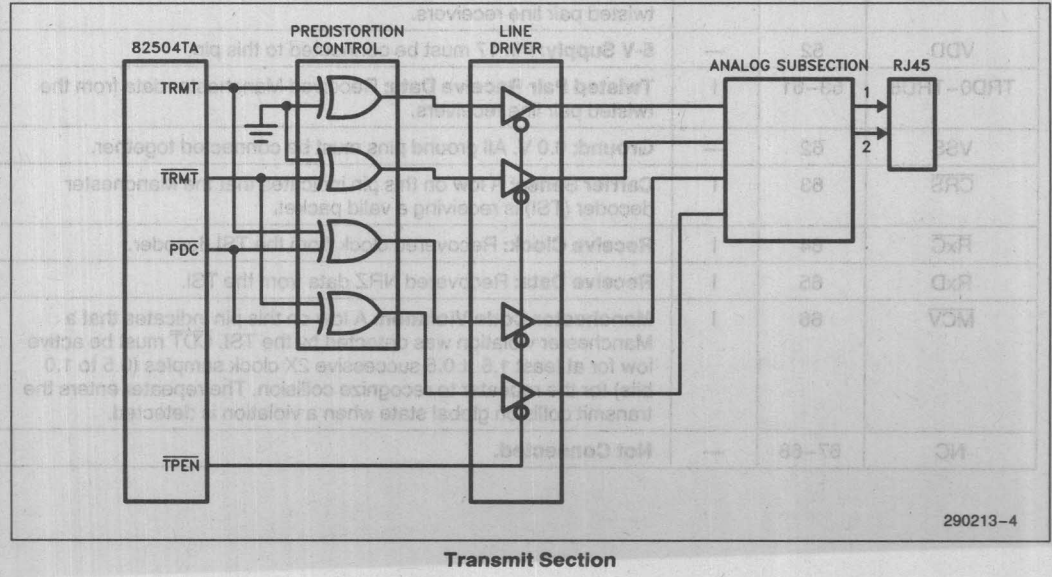
Symbol	Pin No.	Type	Name and Function
NC	1	—	**Not Connected.**
$\overline{\text{FILL}}$	2	I	**FIFO Fill Level:** This pin controls the number of bits loaded into the internal or external FIFO before the bits are unloaded. When low, the fill level is seven. When high, the fill level is eight. This pin is connected to an internal pull-up device.
NC	3–7	—	**Not Connected.**
TESTO0 TESTO1	8 9	O O	**Device Test Outputs:** These pins are used in conjunction with $\overline{\text{TEST0}}$ and $\overline{\text{TEST1}}$ to facilitate device testing. During normal operation these pins are not connected.
VSS	10	—	**Ground:** 0.0V. All ground pins must be connected together.
PD0–PD3	11–14	O	**Port to Disable:** Address of port to disable when traffic is received.
$\overline{\text{PDCTL}}$	15	O	**Port Disable Control:** When low it indicates that the port selected by PD0–PD3 is to be disabled. When high it indicates that all ports should be enabled. The Port Disable address is invalid when PDCTL is high. This pin remains low as long as the Port Disable address pins are valid.
L0	16	O	**LED Status Indicator:** Part of address bus for LED status indicators.
VCC	17	—	**5-V Supply:** All VCC pins must be connected together.
L1–L3	18–20	O	**LED Status Indicators:** Part of Address bus for LED Status Indicators.
LEDCTL	21	O	**LED Control:** When low it indicates that the LED selected by L0–L3 is turned on. When high it indicates that the LED is turned off.
LEDSTRB	22	O	**LED Strobe:** This pin pulses low when the LED address and control pins are valid and an LED status is updated.
CLKOUT	23	O	**20-MHz TTL Clock Output:** This pin is a buffered version of CLK.
VSS	24	—	**Ground:** 0.0 V. All ground pins must be connected together.
TPS	25	O	**Twisted Pair Sense:** This pin indicates presence of carrier to the TSI. It is high while valid Manchester data is being received. If the repeater is sending jam, or is blinding inputs, this pin is driven low.
TRxD	26	O	**TSI Received Data:** Manchester data from the repeater front-end to the Manchester decoder.
$\overline{\text{TRMT}}$	27	O	**Transmit Output:** Retimed Manchester complement to all ports (including the AUI port).
TRMT	28	O	**Transmit Output:** Retimed Manchester data to all ports (including the AUI port).
$\overline{\text{PDC}}$	29	O	**Predistortion Control:** Active low. This signal is used to reduce jitter in a twisted pair environment by preventing overcharge. This pin is asserted for the first 50 ns of any pulse on the TRMT pair. This allows the T-P line drivers to reduce their output voltage during the last 50 ns of 100-ns Manchester pulses. $\overline{\text{PDC}}$ will not produce glitches during consecutive 50-ns pulses.

PIN DESCRIPTIONS (Continued)

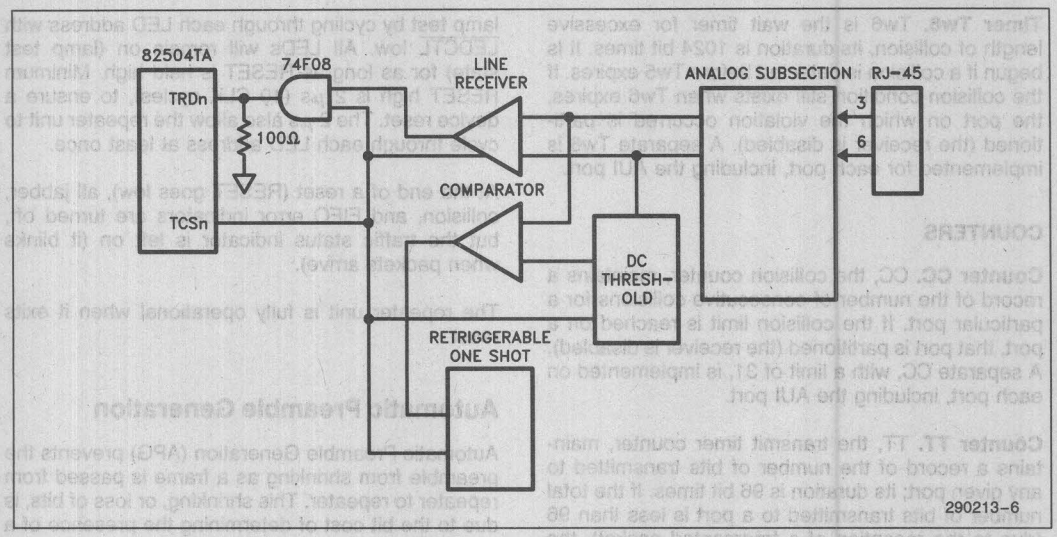
Symbol	Pin No.	Type	Name and Function
$\overline{\text{TPEN}}$	30	O	**Transmit Port Enable:** $\overline{\text{TPEN}}$ is low when TRMT and $\overline{\text{TRMT}}$ contain valid data, jam, or IDL.
VSS	31	—	**Ground:** 0.0 V. All ground pins must be connected together.
CLK	32	I	**System Clock:** 20-MHz ±0.01%, 50% nominal, 40/60% worst-case duty cycle.
$\overline{\text{TEST0}}$ $\overline{\text{TEST1}}$	33 34	I I	**Device Test Control:** If either pin is low, internal test circuitry is enabled to facilitate device testing. If both pins are high the repeater operates normally. These pins are connected to internal pull-up devices.
RESET	35	I	**Device Reset (Schmitt Input):** A high on the pin causes the device to reset. RESET must be low for normal operation.
AUICRS	36	I	**AUI Carrier Sense (Schmitt input):** A high on this pin indicates the presence of a carrier on the AUI port. AUICRS must be active high for at least 2.5 ±0.5 successive 2X clock samples (1 to 1.5 bits) for the repeater to recognize valid AUI carrier.
TCS0-TCS10	37–47	I	**Twisted Pair Carrier Sense (Schmitt input):** A high on any of these pins indicates the presence of carrier on that port. TCS*n* must be active high for at least 2.5 ±0.5 successive 2X clock samples (1 to 1.5 bits) for the repeater to recognize valid T-P carrier.
$\overline{\text{AUICDT}}$	48	I	**AUI Collision Detected:** A low on this pin indicates the presence of a collision at the AUI port. $\overline{\text{AUICDT}}$ must be active low for at least 1.5 ±0.5 successive 2X clock samples (0.5 to 1.0 bits) for the repeater to recognize valid AUI collision.
AUIRxD	49	I	**AUI Receive Data:** Received Manchester data from the AUI line receiver.
TRD9–TRD10	50–51	I	**Twisted Pair Receive Data:** Received Manchester data from the twisted pair line receivers.
VDD	52	—	**5-V Supply:** Pin 17 must be connected to this pin.
TRD0–TRD8	53–61	I	**Twisted Pair Receive Data:** Received Manchester data from the twisted pair line receivers.
VSS	62	—	**Ground:** 0.0 V. All ground pins must be connected together.
$\overline{\text{CRS}}$	63	I	**Carrier Sense:** A low on this pin indicates that the Manchester decoder (TSI) is receiving a valid packet.
$\overline{\text{RxC}}$	64	I	**Receive Clock:** Recovered clock from the TSI decoder.
RxD	65	I	**Receive Data:** Recovered NRZ data from the TSI.
$\overline{\text{MCV}}$	66	I	**Manchester Code Violation:** A low on this pin indicates that a Manchester violation was detected by the TSI. $\overline{\text{CDT}}$ must be active low for at least 1.5 ±0.5 successive 2X clock samples (0.5 to 1.0 bits) for the repeater to recognize collision. The repeater enters the transmit collision global state when a violation is detected.
NC	67–68	—	**Not Connected.**



Typical System Configuration



Transmit Section



Receive Section: Port *n*

FUNCTIONAL DESCRIPTION

The Use of State Diagrams

State diagrams are used throughout the Functional Description section of this data sheet to facilitate concise descriptions. The state diagrams are modelled after—and use the same terminology—those used in the CSMA/CD repeater standard (ANSI/IEEE Std 802.3c-1988). Each state diagram is assumed to represent an independent process; each process communicates by using interprocess flags. Furthermore, the state diagrams are intended to convey the external operation of the MPR, they do not necessarily describe the internal circuitry. For example, the state diagrams imply independent Transmit timers for each port, while in fact, only one timer is used.

The 82505TA state diagrams adhere to the IEEE standard as closely as possible; however, several departures from the standard have been made to account for the inclusion of some of the Twisted Pair MAU's internal functions. Should the state diagrams conflict with the text in this section, the state diagrams should be given preference.

TIMERS

Several timers and counters are implemented in the 82505TA MPR, they are described in the following two sections.

Timer Tw1. Tw1 is the wait timer for the end-of-transmit recovery time (blinding timer), its duration is eight bit times. When the repeater finishes transmitting a packet, Tw1 prevents the repeater from receiving that transmission as a new activity.

Timer Tw2. Tw2 is the wait timer for the end-of-carrier recovery time, its duration is three bit times. When $\overline{\text{AUICDT}}$ is detected making a positive transition, Tw2 prevents the repeater from prematurely detecting the real end-of-collision signal.

Timer Tw3. Tw3 is the wait timer for length of continuous output, its duration is 65,536 bit times. It is started when transmission of a packet begins. If Tw3 expires before transmission of the packet is completed the repeater enters the MAU jabber lockup-protection condition, and interrupts the transmission for a period equal to Tw4.

Timer Tw4. Tw4 is the wait timer for time to disable output for jabber lockup protection, its duration is 96 bit times. When Tw4 is active, transmission to all ports is suspended until the timer expires. The MAU lockup LED is turned on at the next LED counter interval, this indicates that transmission is suspended.

Timer Tw5. Tw5 is the wait timer for length of packet without collision, its duration is 512 bit times. It is started when a port becomes active. If a collision is detected before Tw5 expires, the collision count for that port is augmented, and the port Tw6 is begun. A separate Tw5 is implemented for each port, including the AUI port.

Timer Tw6. Tw6 is the wait timer for excessive length of collision, its duration is 1024 bit times. It is begun if a collision is detected before Tw5 expires. If the collision condition still exists when Tw6 expires, the port on which the violation occurred is partitioned (the receiver is disabled). A separate Tw6 is implemented for each port, including the AUI port.

COUNTERS

Counter CC. CC, the collision counter, maintains a record of the number of consecutive collisions for a particular port. If the collision limit is reached on a port, that port is partitioned (the receiver is disabled). A separate CC, with a limit of 31, is implemented on each port, including the AUI port.

Counter TT. TT, the transmit timer counter, maintains a record of the number of bits transmitted to any given port; its duration is 96 bit times. If the total number of bits transmitted to a port is less than 96 (due to the reception of a fragmented packet), the repeater will enter the receive collision global state and will transmit jam until TT expires, thus extending the frame to $\geq$96 bits.

The TT counter is cleared when the repeater enters the transmit collision global state. This ensures that at least 96 bits of jam are transmitted to all ports before the repeater enters the one remaining port, or blind states from the transmit collision state.

Global State Machine

A single global state machine is implemented for the MPR. The state diagram assumes multiple twisted pair ports and one AUI port.

Auto-Partition and Reconnection

The optional auto-partition and reconnection algorithm described in SC. 9.6.6.2 of the ANSI/IEEE Std 802.3c-1988 is implemented in the 82505TA MPR chip. Each port, including the AUI port, is provided with an individual partition state machine. The state machine for the AUI port corresponds to that described in the standard. The machines for the T-P ports have been modified to reflect the inclusion of several MAU functions; the operation of the machine at the MDI interface remains unchanged.

RESET

The repeater unit is reset when RESET (pin 35) is asserted high. When the 82505TA is reset, the repeater unit disconnects all ports and performs a lamp test by cycling through each LED address with $\overline{\text{LEDCTL}}$ low. All LEDs will remain on (lamp test state) for as long as RESET is held high. Minimum RESET high is 2 μs (40 CLK cycles), to ensure a device reset. The 2 μs also allow the repeater unit to cycle through each LED address at least once.

At the end of a reset (RESET goes low), all jabber, collision, and FIFO error indicators are turned off, but the traffic status indicator is left on (it blinks when packets arrive).

The repeater unit is fully operational when it exits reset.

Automatic Preamble Generation

Automatic Preamble Generation (APG) prevents the preamble from shrinking as a frame is passed from repeater to repeater. This shrinking, or loss of bits, is due to the bit cost of determining the presence of a carrier and synchronizing the Manchester data for NRZ data and clock recovery.

The APG circuit compensates for the bit loss by beginning transmission of new preamble bits before the FIFO limit is reached. When bits from the incoming frame reach the FIFO limit they are synchronously switched into the awaiting pretransmitted preamble. If the logic polarity of the first bit out of the FIFO is not the value expected, an extra preamble bit is transmitted. This prevents corruption of the preamble pattern when the transmitted bit stream is switched from the APG generator to the FIFO data.

The delay between carrier detection and start of APG depends on the FIFO fill level selected, and the type of active port.

The delay for a T-P port is such that the number of preamble bits added by the repeater is equal to the number of bits ($\pm$1 bit) lost while detecting the frame. That is, the latency of bits through the repeater equals the delay of preamble start introduced by the repeater. If the FIFO fill level ($\overline{\text{FILL}}$) is changed, the delay for start of APG is automatically changed to compensate for the new latency of the FIFO.

For the AUI port, the bit loss in detecting the frame can range from one to eight bits. Therefore, the beginning of APG caused by the AUI port is dependent only a a carrier detection, and is not delayed. This allows the repeater unit to recover a maximum of three bits lost by the attached MAU in frame detection.

The leading edge of the first preamble bit transmitted by the repeater (as seen on the line) indicates the beginning of a 100-ns positive voltage (TTL logic 1).

LED STATUS INDICATOR ADDRESSES

L3	L2	L1	L0	Address	Description	Asserted LED State
0	0	0	0	0	TPP 0 Partition Jabber	On
0	0	0	1	1	TPP 1 Partition Jabber	On
0	0	1	0	2	TPP 2 Partition Jabber	On
0	0	1	1	3	TPP 3 Partition Jabber	On
0	1	0	0	4	TPP 4 Partition Jabber	On
0	1	0	1	5	TPP 5 Partition Jabber	On
0	1	1	0	6	TPP 6 Partition Jabber	On
0	1	1	1	7	TPP 7 Partition Jabber	On
1	0	0	0	8	TPP 8 Partition Jabber	On
1	0	0	1	9	TPP 9 Partition Jabber	On
1	0	1	0	A	TPP 10 Partition Jabber	On
1	0	1	1	B	AUI Partition Jabber	On
1	1	0	0	C	FIFO Error	On
1	1	0	1	D	Traffic	Off
1	1	1	0	E	Jam*	On
1	1	1	1	F	MAU Lockup Protection	On

* This indicates the transmission of jam for collision, packet fragments, and FIFO errors.

LED Controller

The LED controller contains a 21-bit counter (105 ms at 20 MHz). At each 105-ms interval, the controller loads the status of each event into a shift register and shifts the status out as $\overline{\text{LEDCTL}}$. If the port indicates a change in status from the previous interval, the appropriate LED is toggled. Thus, for each event, the minimum time an LED is on or off is 105 ms, and the LED will not change state until an additional 105 ms have elapsed.

The status indicators, with the exception of the jabber indicators (address 0 to 11), have a 50% duty cycle when they are asserted (105 ms on, 105 ms off). The jabber indicators will remain on for as long as the affected ports are partitioned (Receive disabled).

The following table shows the addressing used for the LED status indicators. An LED is turned on by a negative pulse on $\overline{\text{LEDSTRB}}$ when $\overline{\text{LEDCTL}}$ is low. An LED is turned off by a negative pulse on $\overline{\text{LEDSTRB}}$ when $\overline{\text{LEDCTL}}$ is high.

A status LED interface using two 74LS259 addressable latches is shown in the following figure.

The upper LED address bit (L3) is gated with $\overline{\text{LEDSTRB}}$ to provide the strobe signal to each 74LS259 device. The CLEAR input is tied high since the repeater will initialize the latches during the LED lamp test.

Port Disable Controller

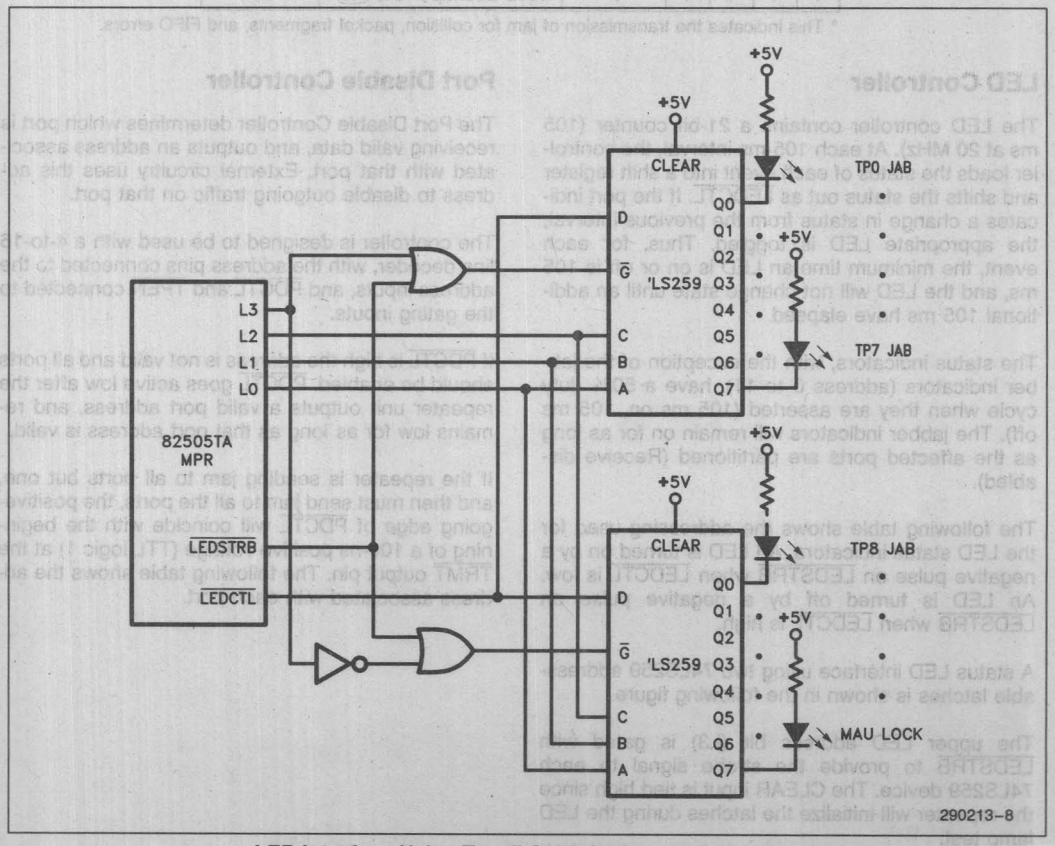
The Port Disable Controller determines which port is receiving valid data, and outputs an address associated with that port. External circuitry uses this address to disable outgoing traffic on that port.

The controller is designed to be used with a 4-to-16 line decoder, with the address pins connected to the address inputs, and $\overline{\text{PDCTL}}$ and $\overline{\text{TPEN}}$ connected to the gating inputs.

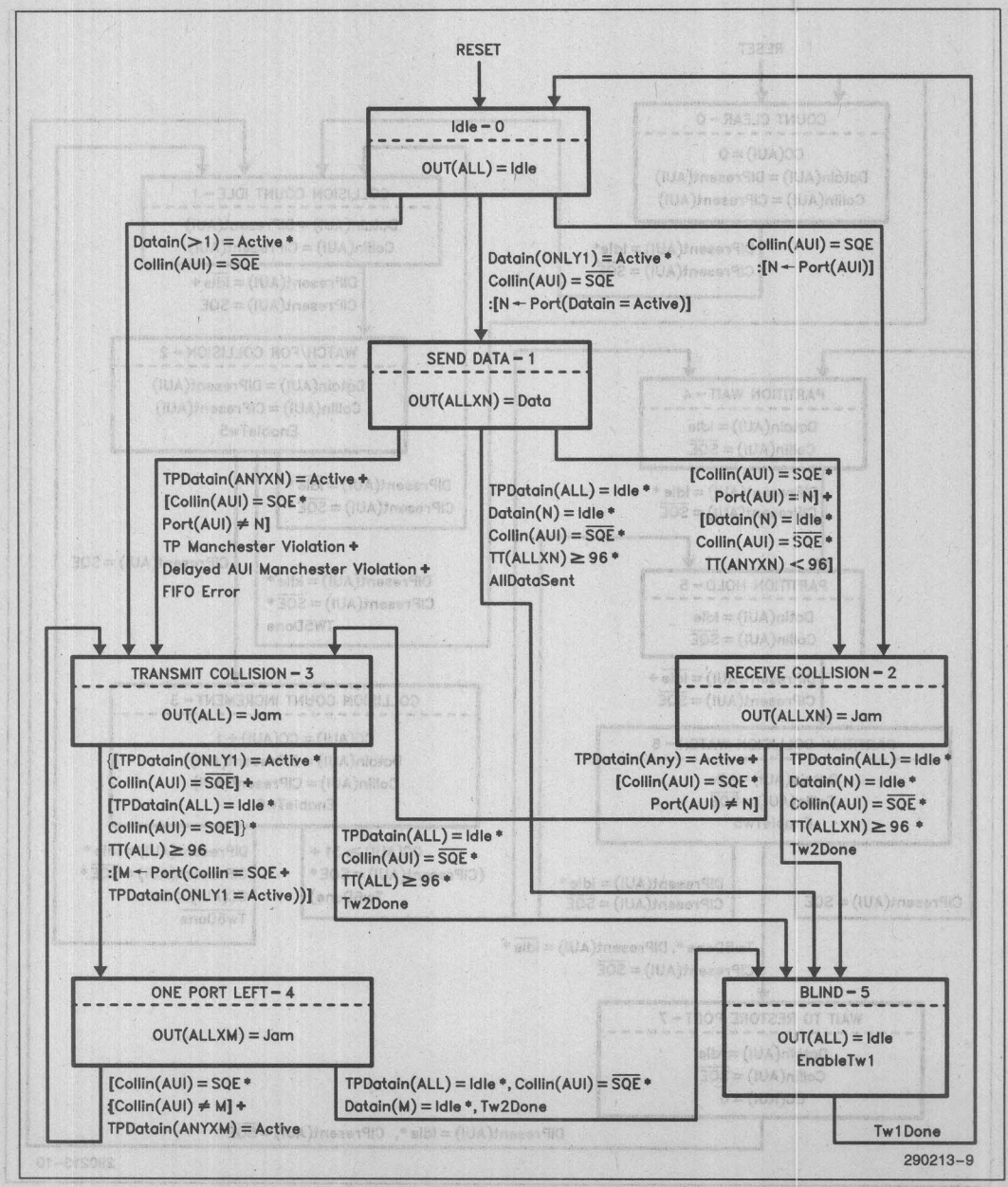
If $\overline{\text{PDCTL}}$ is high the address is not valid and all ports should be enabled. $\overline{\text{PDCTL}}$ goes active low after the repeater unit outputs a valid port address, and remains low for as long as that port address is valid.

If the repeater is sending jam to all ports but one, and then must send jam to all the ports, the positive-going edge of $\overline{\text{PDCTL}}$ will coincide with the beginning of a 100-ns positive voltage (TTL logic 1) at the $\overline{\text{TRMT}}$ output pin. The following table shows the address associated with each port.

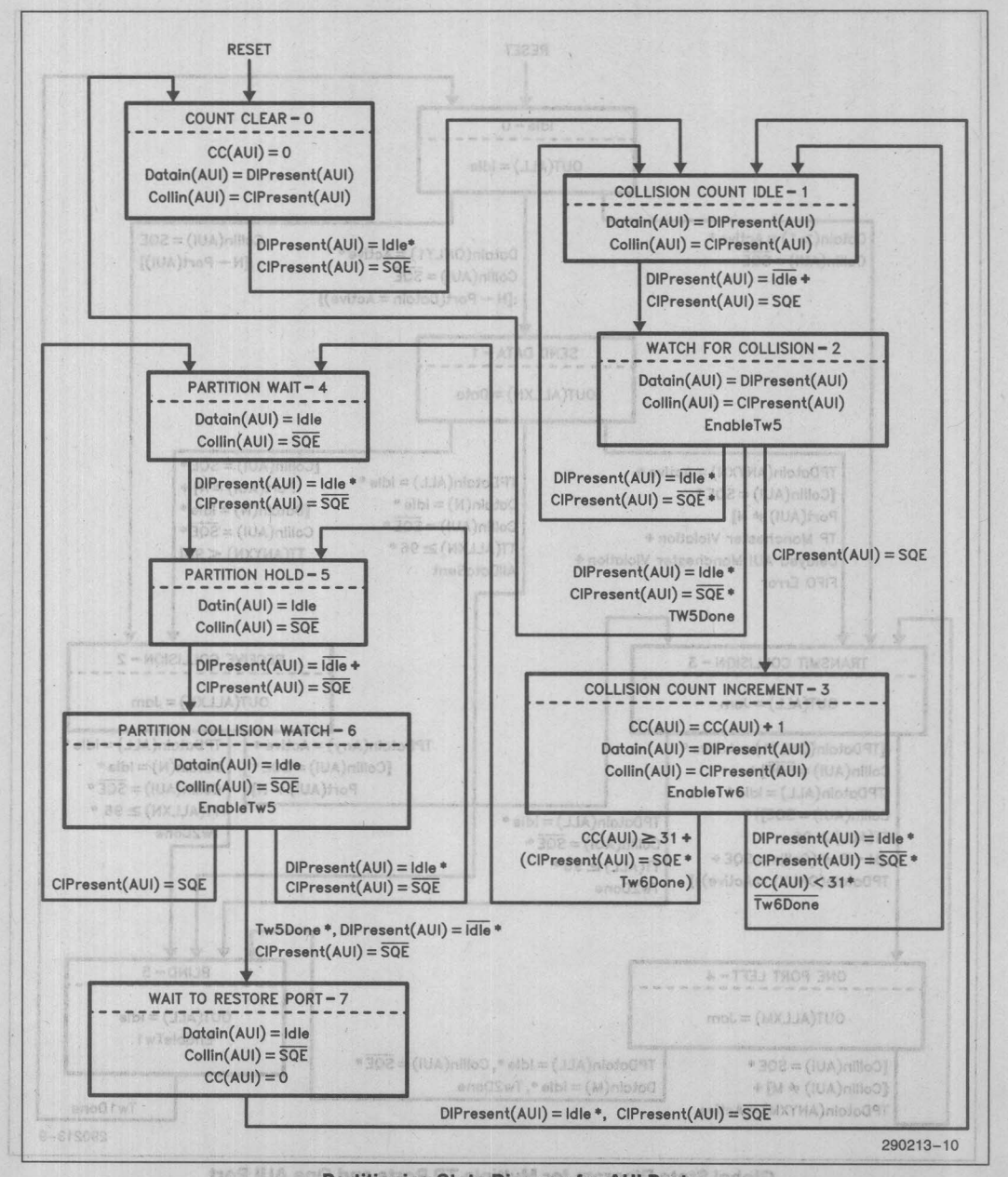
1	1	0	0	C	Not Assigned
1	1	0	1	D	Not Assigned
1	1	1	0	E	Not Assigned
1	1	1	1	F	Not Assigned



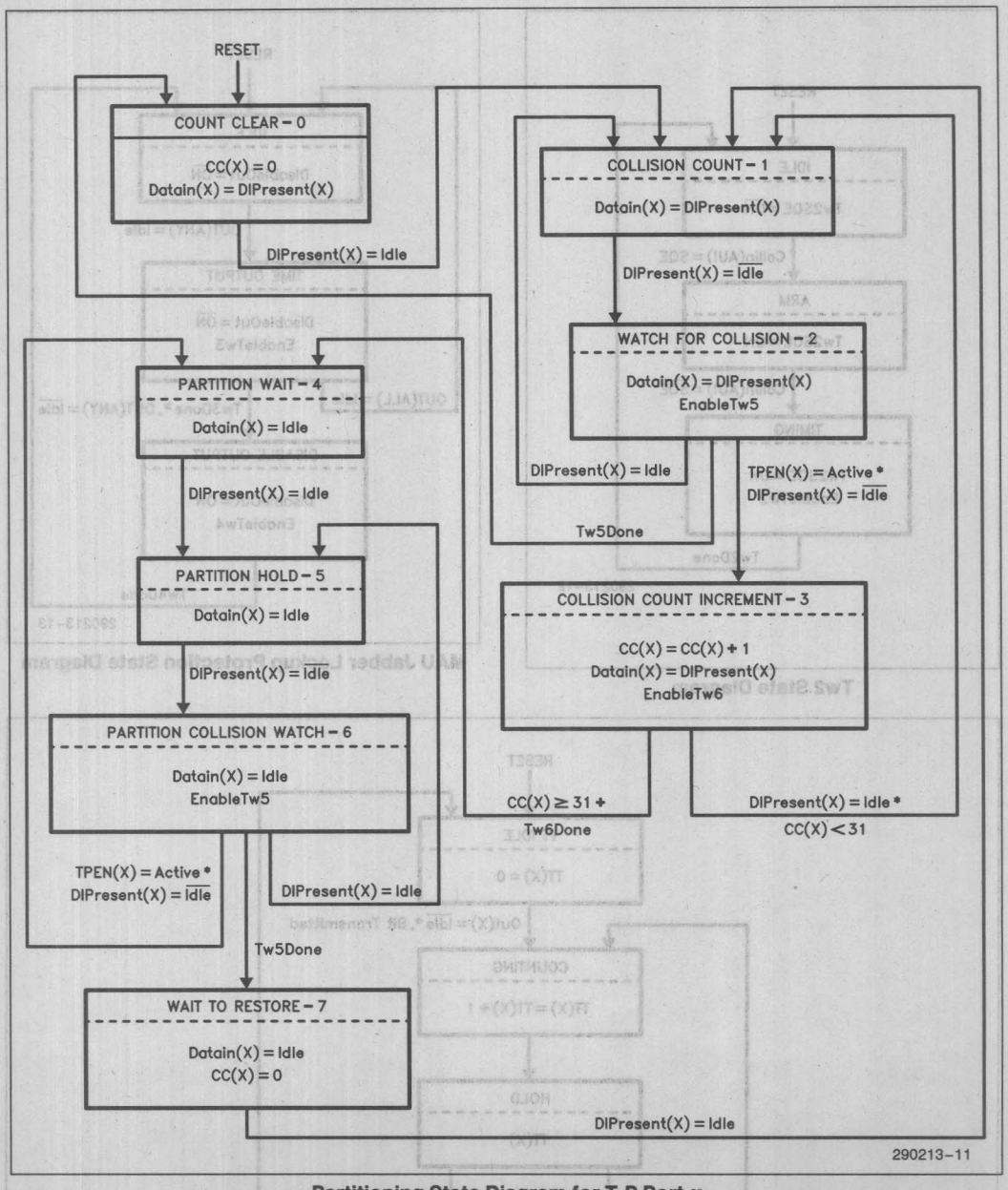
LED Interface Using Two 'LS259 Addressable Latches



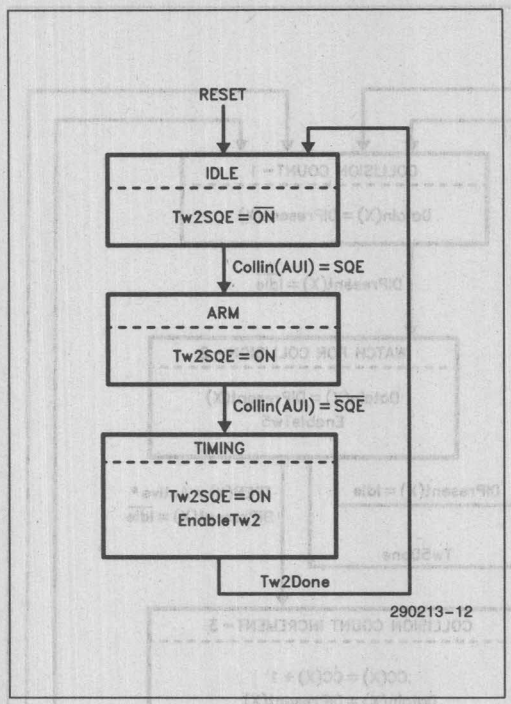
Global State Diagram for Multiple TP Ports and One AUI Port



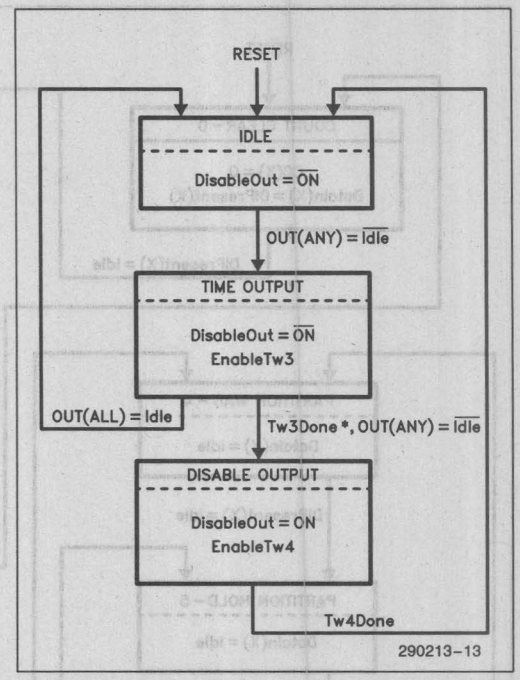
Partitioning State Diagram for AUI Port



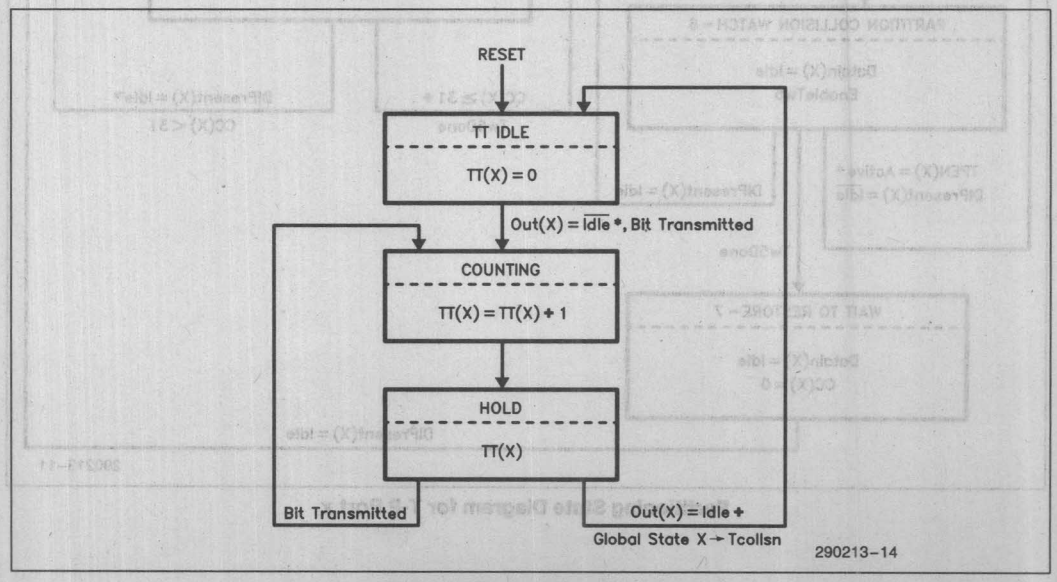
Partitioning State Diagram for T-P Port *x*



Tw2 State Diagram



MAU Jabber Lockup Protection State Diagram



Transmit Timer State Diagram

CHARACTERISTICS

DC Characteristics

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	—
V_{IH}	Input High Voltage	2.0	V_{DD} + 0.5	V	—
V_{OL}	Output Low Voltage	—	0.5	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage	2.4	—	V	I_{OH} = −0.4 mA
I_{CC}	Power Supply Current	—	75	mA	V_{CC} = 5.5V
I_{LI}	Input Leakage Current (TTL)	—	10	μA	V_{IH} = 5.5V
I_{LI}	Schmitt Inputs	—	10	μA	V_{IH} = 5.5V
I_{LI}	Inputs with Pull-Up	—	500	μA	V_{IH} = 5.5V
PD	Power Dissipation (25°C)	—	0.33	W	V_{CC} = 5.0V
PD	Power Dissipation (0°C)	—	0.4	W	V_{CC} = 5.0V

1

ABSOLUTE MAXIMUM RATING*

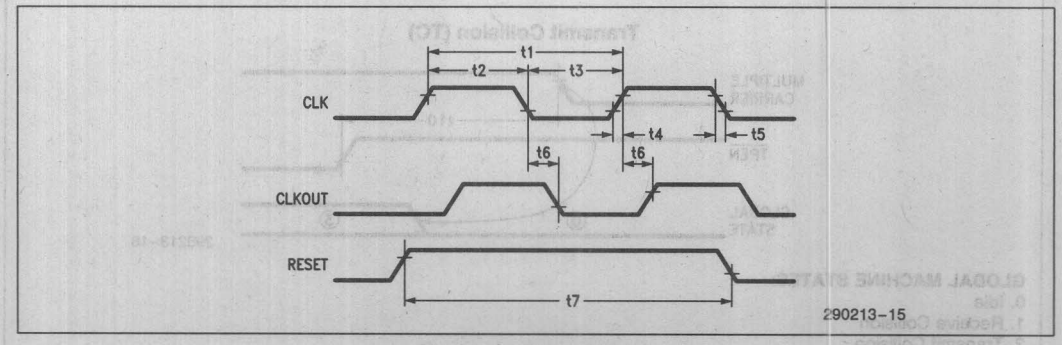
Ambient Operating Temperature (Ta) . . . . . 0°C, 70°C
Storage Temperature . . . . . −40°C, +125°C
Power Dissipation . . . . . 400 mW
Voltage On Any Pin
with Respect to Ground . . . . . −0.5V, V_{CC} + 0.5V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

CLOCK AND RESET TIMINGS

Symbol	Description	Min	Max	Unit
t1	Clock period	49.995	50.005	ns
t2	Clock high	20	30	ns
t3	Clock low	20	30	ns
t4	Clock rise time	—	10	ns
t5	Clock fall time	—	10	ns
t6	CLKOUT propagation delay	10	16	ns
t7	RESET pulse width	40	—	CLK

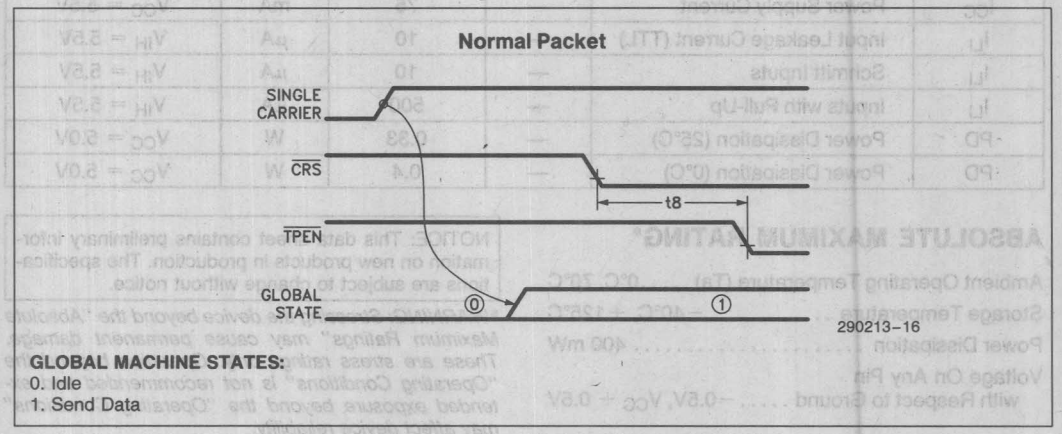


Clock and Reset Timing

TRANSMIT TIMINGS

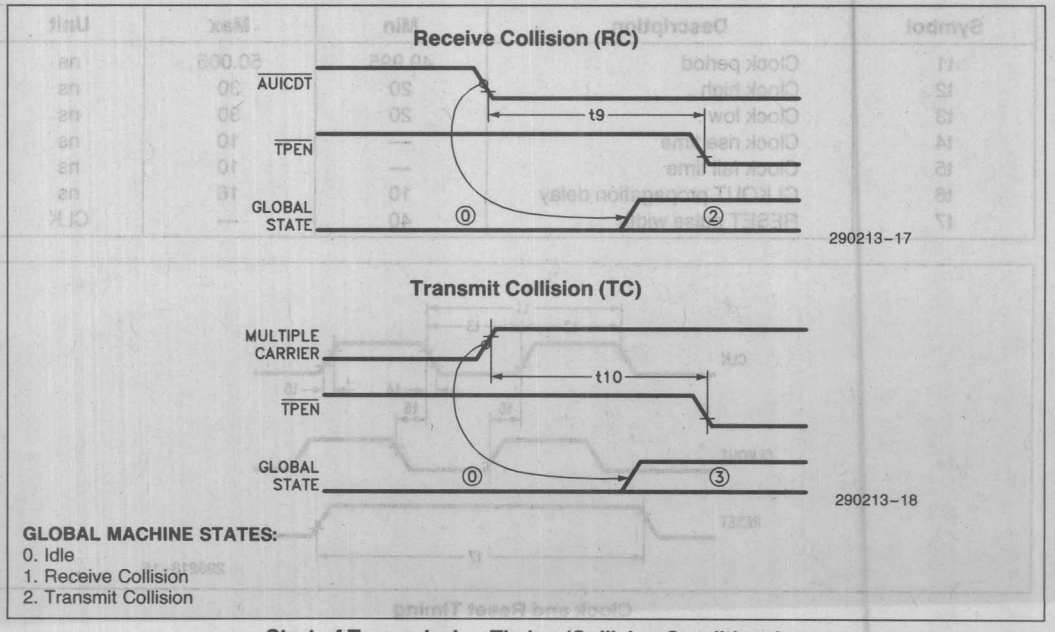
Symbol	Description	Min	Max	Unit
t8	Delay from $\overline{CRS}$ low to $\overline{TPEN}$ low.			
	T-P Port, $\overline{FILL}$ = 0 (FIFO fill = 7 bits)	5	6	bits
	T-P Port $\overline{FILL}$ = 1, (FIFO fill = 8 bits)	6	7	bits
	AUI Port, $\overline{FILL}$ = x (FIFO fill = don't care)	4.5	5	bits
t9	$\overline{AUICDT}$ low to $\overline{TPEN}$ low	4	5	bits
t10	Multiple Carrier* to $\overline{TPEN}$ low	5	6	bits

* Carrier is any of TCSX or AUICRS



GLOBAL MACHINE STATES:
0. Idle
1. Send Data

Start of Transmission Timing (Normal Packet)

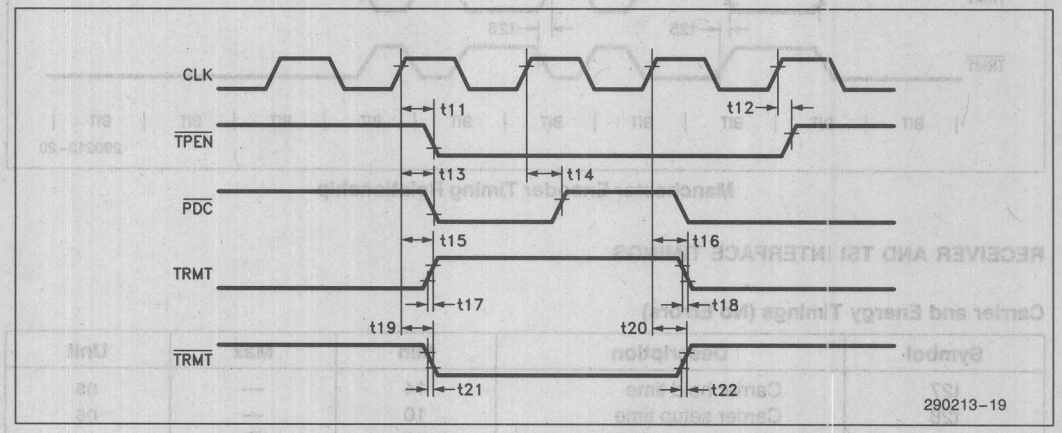


GLOBAL MACHINE STATES:
0. Idle
1. Receive Collision
2. Transmit Collision

Start of Transmission Timing (Collision Conditions)

Manchester Encoder Timing

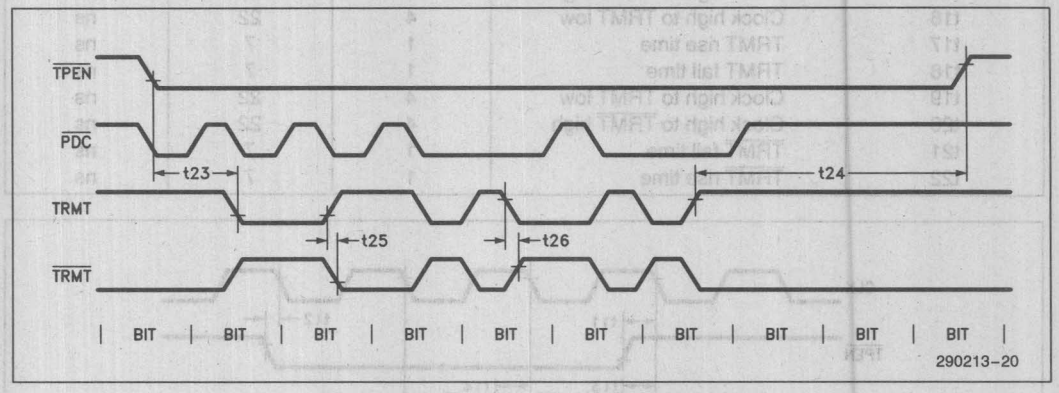
Symbol	Description	Min	Max	Unit
t11	Clock high to $\overline{\text{TPEN}}$ low	4	22	ns
t12	Clock high to $\overline{\text{TPEN}}$ high	4	22	ns
t13	Clock high to $\overline{\text{PDC}}$ low	4	22	ns
t14	Clock high to $\overline{\text{PDC}}$ high	4	22	ns
t15	Clock high to TRMT high	4	22	ns
t16	Clock high to TRMT low	4	22	ns
t17	TRMT rise time	1	7	ns
t18	TRMT fall time	1	7	ns
t19	Clock high to $\overline{\text{TRMT}}$ low	4	22	ns
t20	Clock high to $\overline{\text{TRMT}}$ high	4	22	ns
t21	$\overline{\text{TRMT}}$ fall time	1	7	ns
t22	$\overline{\text{TRMT}}$ rise time	1	7	ns



Manchester Encoder Timing

Manchester Encoder Timing Relationships

Symbol	Description	Min	Max	Unit
t23	$\overline{\text{TPEN}}$ low to TRMT low	1	—	bits
t24	TRMT high to $\overline{\text{TPEN}}$ high	2.5	3.0	bits
t25	TRMT high to $\overline{\text{TRMT}}$ low	0	5	ns
t26	TRMT low to $\overline{\text{TRMT}}$ high	0	5	ns

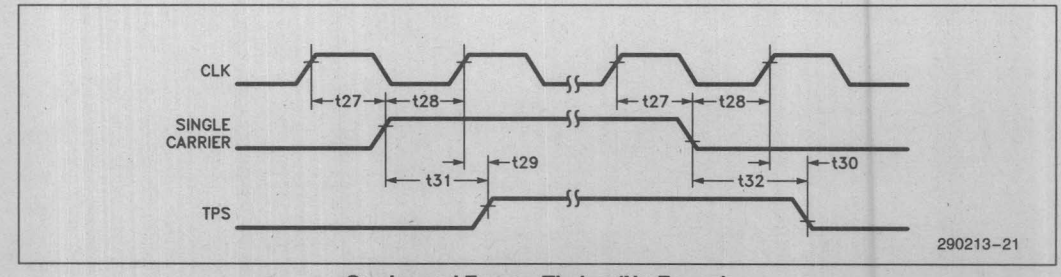


Manchester Encoder Timing Relationship

RECEIVER AND TSI INTERFACE TIMINGS

Carrier and Energy Timings (No Errors)

Symbol	Description	Min	Max	Unit
t27	Carrier hold time	14	—	ns
t28	Carrier setup time	10	—	ns
t29	Clock high to TPS high	20	30	ns
t30	Clock high to TPS low	20	30	ns
t31	Carrier high to TPS high	4	5	CLK
t32	Carrier low to TPS low	2	3	CLK

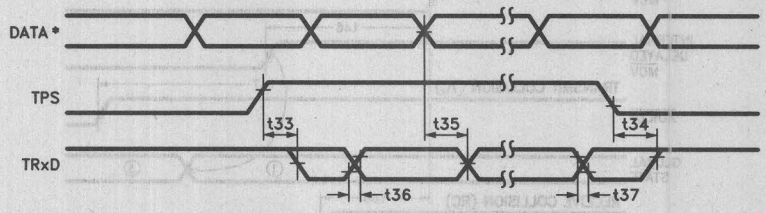


Carrier and Energy Timing (No Errors)

Data and TRMT Timings

Symbol	Description	Min	Max	Unit
t33	TPS high to TRXD valid	5	12	ns
t34	TPS low to TRXD high	5	12	ns
t35	Data† to TRSD delay	5	35	ns
t36	TRXD rise time	1	7	ns
t37	TRXD fall time	1	7	ns

†Data refers to the data signal (either TRDx or AUIRxD) from the active receiving port.

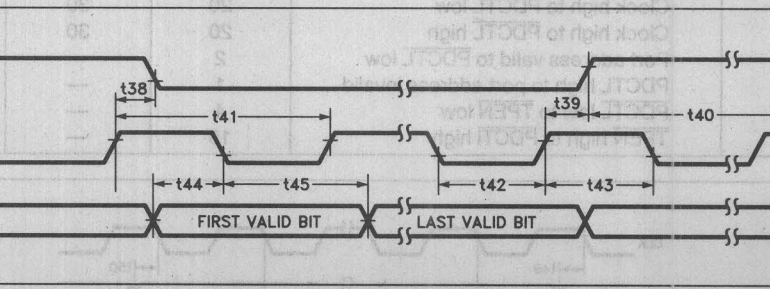


*Data refers to the data signal (either TRDx or AUIRXD) from the active port.

Data and MRXD Timing

Decoded NRZ Data Timings

Symbol	Description	Min	Max	Unit
t38	$\overline{RxC}$ high to $\overline{CRs}$ low	5	19	ns
t39	$\overline{RxC}$ high to $\overline{CRS}$ high	5	19	sn
t40	Time between $\overline{CRS}$ low	16	—	bits
t41	$\overline{RxC}$ period	78	—	ns
t42	$\overline{RxC}$ low	30	—	ns
t43	$\overline{RxC}$ high	36	—	ns
t44	RXD setup time	40	—	ns
t45	RXD hold time	30	—	ns

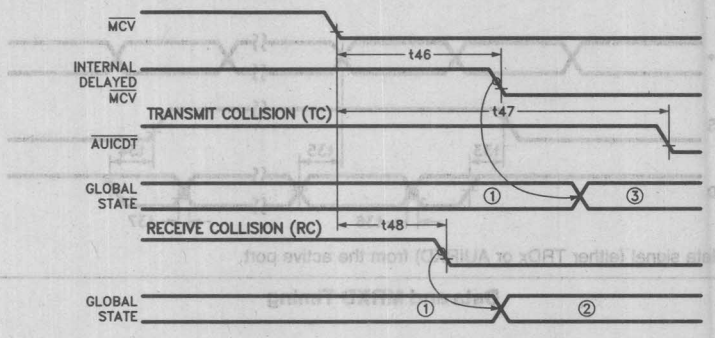


Decoded NRZ Data Timing

AUI COLLISION INTERFACE

$\overline{CDT}$ and $\overline{AUICDT}$ Timing Relationship for AUI Port

Symbol	Description	Min	Max	Unit
t46	Delay for $\overline{MCV}$ low	10.5	11	bits
t47	$\overline{MCV}$ low to $\overline{AUICDT}$ low for T-COLLSN	9.5	—	bits
t48	$\overline{MCV}$ low to $\overline{AUICDT}$ low for R-COLLSN	—	9	bits

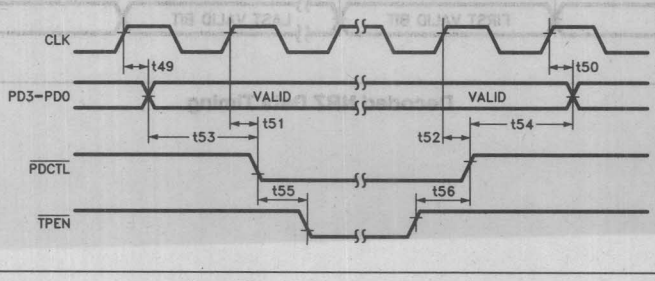


GLOBAL MACHINE STATES:
1. Send Data
2. Receive Collision
3. Transmit Collision

$\overline{MCV}$, $\overline{AUICDT}$ Timing Relationship for AUI Port

Port Disable Control Timings

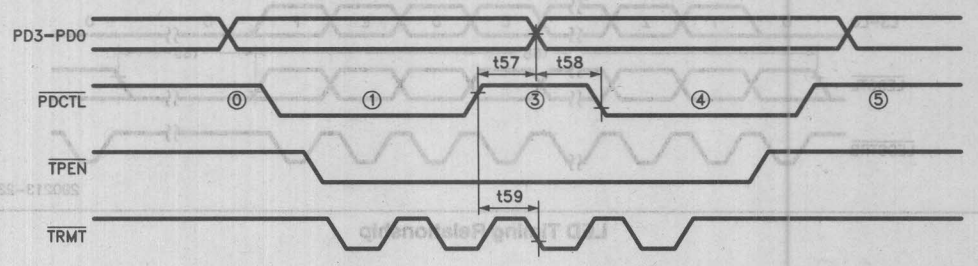
Symbol	Description	Min	Max	Unit
t49	Clock high to port address valid	20	30	ns
t50	Clock high to port address invalid	20	30	ns
t51	Clock high to $\overline{PDCTL}$ low	20	30	ns
t52	Clock high to $\overline{PDCTL}$ high	20	30	ns
t53	Port address valid to $\overline{PDCTL}$ low	2	—	CLK
t54	PDCTL high to port address invalid	1	—	CLK
t55	$\overline{PDCTL}$ low to $\overline{TPEN}$ low	4	—	CLK
t56	$\overline{TPEN}$ high to $\overline{PDCTl}$ high	19	—	CLK



Port Disable Control Timing

Port Disable Timing (One Port Left)

Symbol	Description	Min	Max	Unit
t57	$\overline{\text{PDCTL}}$ high to port address invalid	1	—	CLK
t58	Port address valid to $\overline{\text{PDCTL}}$ low	2	—	CLK
t59	$\overline{\text{PDCTI}}$ high to TRMT low	1	—	bits

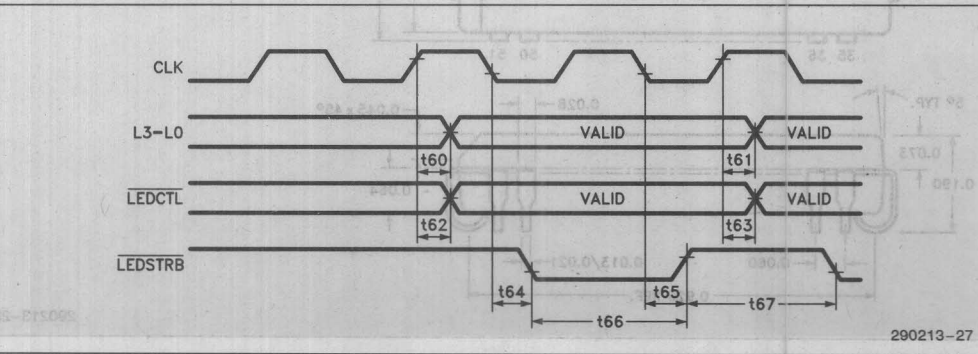


GLOBAL MACHINE STATES:
0. Idle
1. Send Data
3. Transmit Collision
4. One Port Left
5. Blind

Port Disable Timing (One Port Left)

LED Control Timing

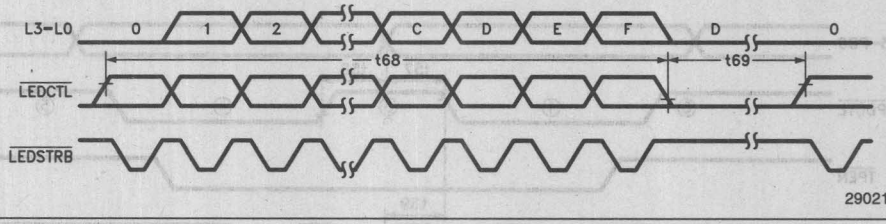
Symbol	Description	Min	Max	Unit
t60	Clock high to address valid	20	30	ns
t61	Clock high to address invalid	20	30	ns
t62	Clock high to $\overline{\text{LEDCTL}}$ valid	20	30	ns
t63	Clock high to $\overline{\text{LEDCTL}}$ invalid	20	30	ns
t64	Clock low to $\overline{\text{LEDSTRB}}$ low	20	30	ns
t65	Clock low to $\overline{\text{LEDSTRB}}$ high	20	30	ns
t66	$\overline{\text{LEDSTRB}}$ low	1	—	CLK
t67	Time between strobes	1	—	CLK



LED Control Timing

LED Timing Relationship

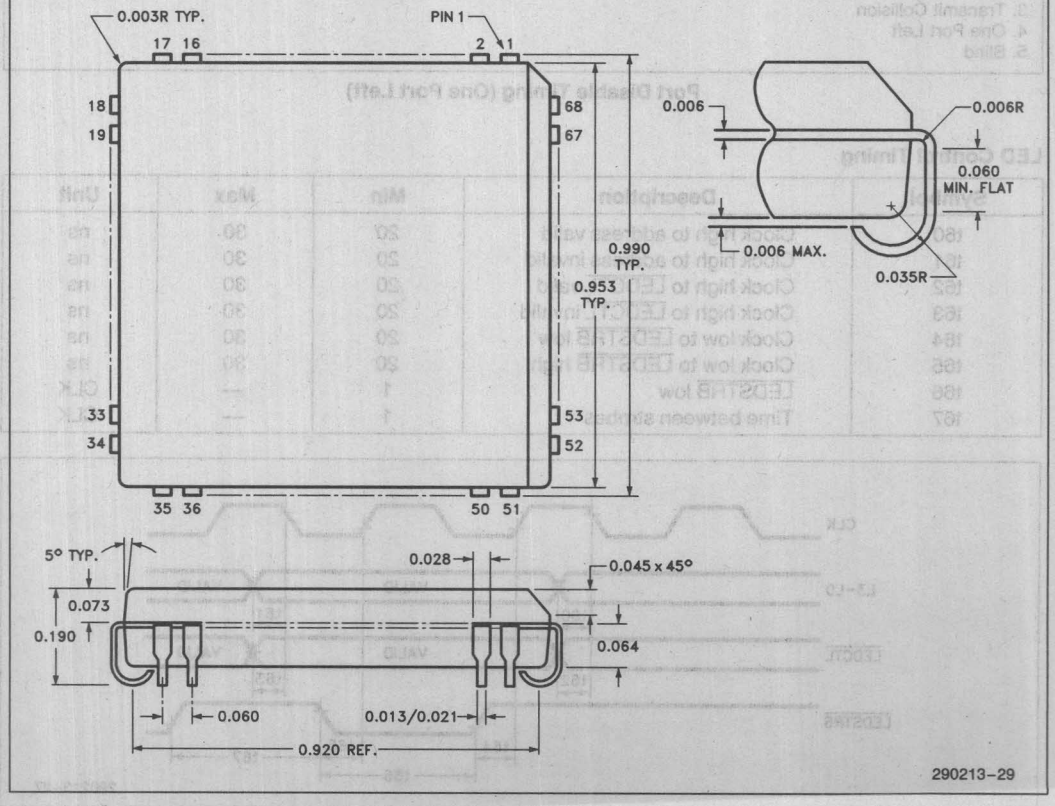
Symbol	Description	Min	Max	Unit
t68	LED update duration	1.6	—	μs
t69	Interoperation time	105	—	ms



LED Timing Relationship

OUTLINE DIAGRAM

Dimensions are in Inches



82506TB
TWISTED PAIR MEDIUM ATTACHMENT UNIT (TP MAU)

- Complies with IEEE 802.3 10BASE-T Draft 11 for Twisted Pair Interface
- Conforms to IEEE 802.3 Standard for Attachment Unit Interface (AUI)
- Direct Interface to AUI and Twisted Pair Isolation Transformers
- On-Chip Line Drivers and Receivers
- LED Drivers for Transmit, Receive, Collision, and Jabber Status
- Generates Internal Predistortion Signal
- Resetable Jabber Function
- Selectable Link Integrity (LI) Function
- Selectable Signal Quality Error (SQE) Function
- Low-Power CMOS Technology
- Single 5-V Supply
- 28-Lead Plastic DIP and SOJ Packages (See Packaging Spec Order No. 231369)

1

The 82506TB Twisted Pair Medium Attachment Unit (TP MAU) is intended for local area network (LAN) designs that interface the IEEE 802.3-1988 AUI cable to the twisted pair wire (10BASE-T). It offers LAN designers a cost-effective, integrated solution to the problem of upgrading existing standard Ethernet* networks to twisted pair. The 82506TB complies with IEEE 802.3 AUI specifications and IEEE 802.3 10BASE-T Draft 11 specifications. The device incorporates the interface circuitry and both the AUI and twisted pair line drivers and receivers in a low-power CMOS package. The 82506TB TP MAU internally generates predistortion signals to eliminate line overcharge and improve jitter performance. It provides selectable 10BASE-T features for simplified network management, including selectable signal quality error (SQE) test, link integrity test, and jabber protection. In addition, the 82506TB TP MAU supports LED status indicators for transmit, receive, jabber, and collision. It is fabricated using CMOS-process technology and is available in 28-lead plastic DIP and 28-lead SOJ packages.

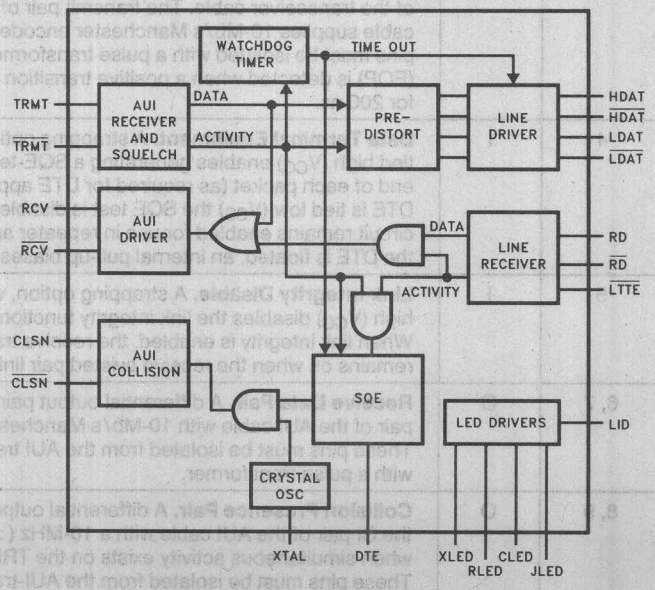


Figure 1. 82506TB TP MAU Block Diagram

Manufactured and tested for Intel by AT&T in accordance with AT&T internal standards.

*Ethernet® is a registered trademark of Xerox Corporation.

© INTEL CORPORATION, 1990
© AT&T, 1990

September 1990
Order Number: 290260-002

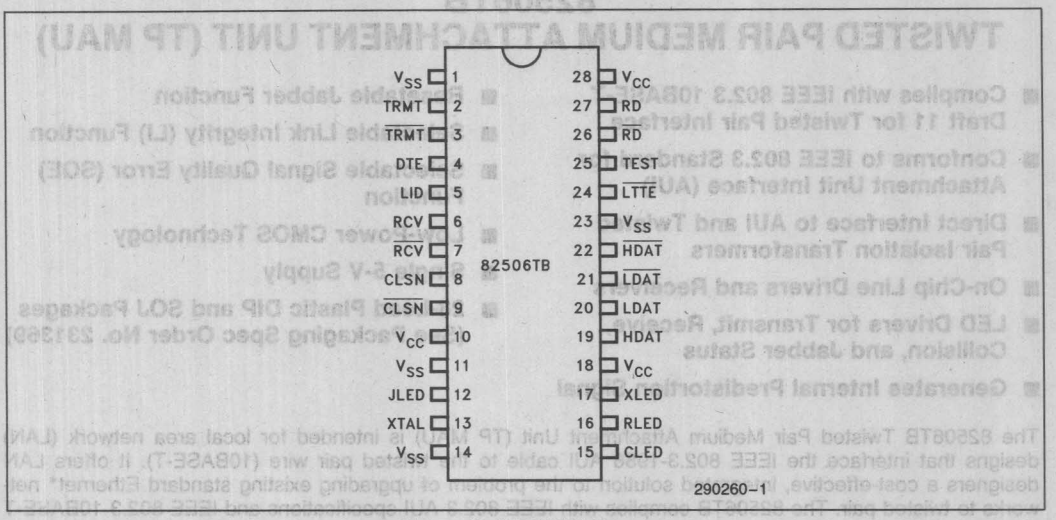


Figure 2. 82506TB Pinout

TABLE 1. 82506TB Pinout Description

Symbol	Pin No.	Type	Name and Function
V_{SS}	1	—	**Analog Ground.**
TRMT, $\overline{\text{TRMT}}$	2, 3	I	**Transmit Data.** A differentially driven input tied to the D0 pair of the transceiver cable. The transmit pair of the transceiver cable supplies 10-Mb/s Manchester encoded data. These pins must be isolated with a pulse transformer. End of Packet (EOP) is detected when a positive transition has not occurred for 200 ns.
DTE	4	I	**Data Terminal Equipment.** A strapping option, which when tied high (V_{CC}) enables generating a SQE-test signal at the end of each packet (as required for DTE applications). When DTE is tied low (V_{SS}) the SQE test is disabled, but the collision circuit remains enabled for use in repeater applications. When the DTE is floated, an internal pull-up biases the signal high.
LID	5	I	**Link Integrity Disable.** A strapping option, which when tied high (V_{CC}) disables the link integrity function of the TPMAU. When link integrity is enabled, the receive traffic indicator remains on when the receive twisted pair link is present.
RCV, $\overline{\text{RCV}}$	6, 7	O	**Receive Data Pair.** A differential output pair that drives the DI pair of the AUI cable with 10-Mb/s Manchester encoded data. These pins must be isolated from the AUI transceiver cable with a pulse transformer.
CLSN, $\overline{\text{CLSN}}$	8, 9	O	**Collision Presence Pair.** A differential output pair that drives the CI pair of the AUI cable with a 10-MHz (±15%) signal when simultaneous activity exists on the TRMT and RD pairs. These pins must be isolated from the AUI-transceiver cable with a pulse transformer.
V_{CC}	10	—	**Power.** Digital, 5 V.
V_{SS}	11	—	**Ground.**
JLED	12	O	**Jabber Indicator.** Indicates that the watchdog timer has timed out and the twisted pair drivers have been disabled.

TABLE 1. 82506TB Pinout Description (Continued)

Symbol	Pin No.	Type	Name and Function
XTAL	13	I	**Crystal In.** A 20-MHz clock input. This signal can be driven by a 20-MHz, parallel-resonant crystal or a MOS level clock with a 60/40 duty cycle.
V_{SS}	14	I	**Ground.** For XTAL (Pin 13) and indicator output drivers.
CLED	15	O	**Collision Indicator.** Indicates that a collision has been detected by the TP medium attachment unit.
RLED	16	O	**Receive Indicator.** Indicates that a reception from the network is in progress.
XLED	17	O	**Transmit Indicator.** Indicates that a transmission onto the network is in progress.
V_{CC}	18	—	**Power.** 5 V.
HDAT, $\overline{\text{HDAT}}$ LDAT, $\overline{\text{LDAT}}$	19, 22 20, 21	O	**TP Transmit Pair Drivers.** These four outputs constitute the twisted-pair drivers, which have predistortion capabilities. The HDAT/$\overline{\text{HDAT}}$ outputs generate the 10-Mb/s Manchester encoded data. The LDAT/$\overline{\text{LDAT}}$ outputs mirror the HDAT/$\overline{\text{HDAT}}$ outputs except for "fat" bit occurrences. During the second half of a "fat" bit (either high or low), the LDAT/$\overline{\text{LDAT}}$ outputs are inverted with respect to HDAT/$\overline{\text{HDAT}}$ outputs. This signal behavior reduces the amount of jitter by preventing overcharge on the twisted-pair medium.
$\overline{\text{LTTE}}$	24	I	**Lower TP Threshold Enable (Active Low).** For 10BASE-T compatible operation this pin must be left open. But, if this pin is grounded, the TP receiver threshold is lowered by approximately 4.5 db from the nominal 10BASE-T required specification. By using this lower-threshold option and by selecting different compensation resistor values for wave-construction circuits, a customized interface is possible for non-10BASE-T applications. However, once this lower threshold is invoked, the wiring used must not be a bundled system (e.g., 25 pair) where other services reside (e.g., voice, other 10BASE-T users, etc.).
TEST	25	I	**Test.** This pin is used for testing; it should be connected to V_{SS} during normal operation.
$\overline{\text{RD}}$, RD	26, 27	I	**TP Receive Pair.** The differential twisted pair receiver. The receive pair is connected to the twisted pair medium and is driven with 10-Mb/s Manchester encoded data.
V_{CC}	28	—	**Analog Power.** 5 V.

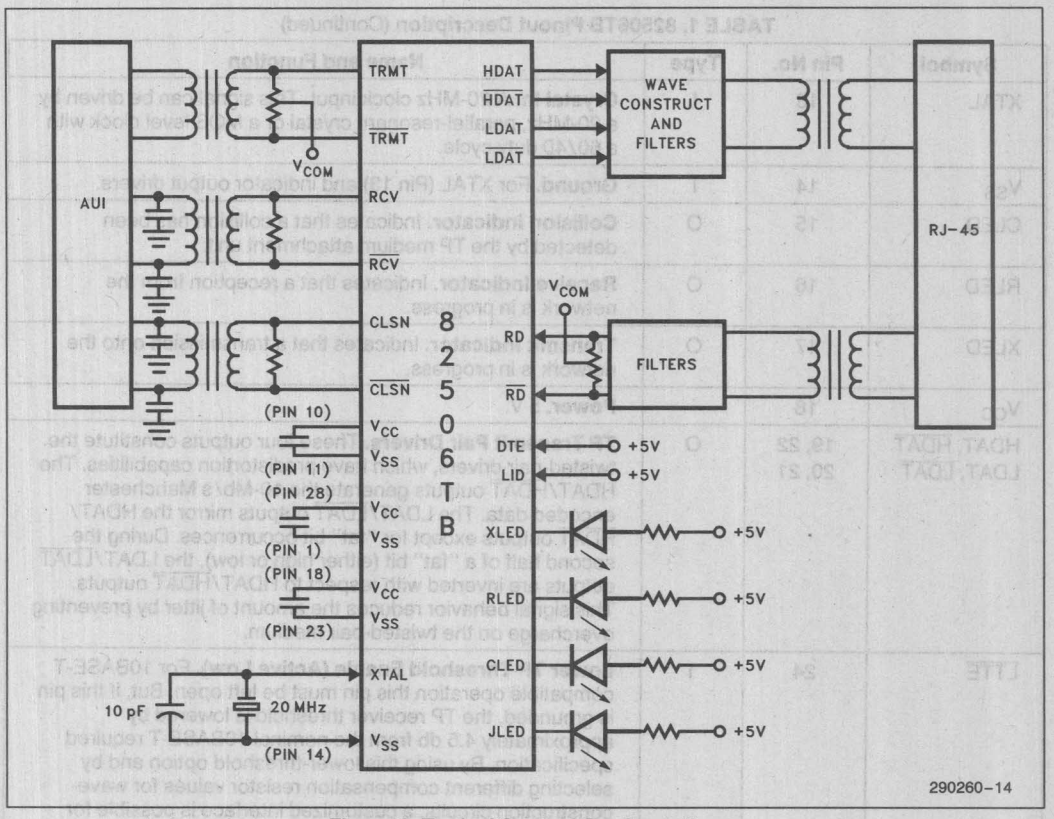
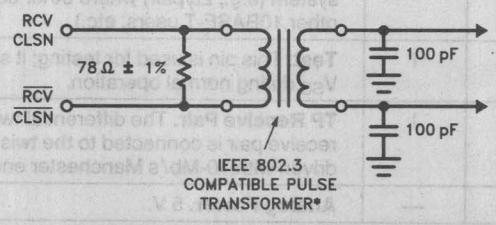


Figure 3. Typical System Configuration



*Required transformers are the Pulse Engineering Inc. (PE64503), TDK Corp. (TLA-100-3E), Coilcraft (LAXET304 or LO323-A), or equivalents.

Figure 4. Typical Load Output on the Outputs of RCV/$\overline{\text{RCV}}$ and CLSN/$\overline{\text{CLSN}}$

FUNCTIONAL DESCRIPTION

Overview

The 82506TB provides the transmit, receive, and collision detection functions specified by the IEEE 802.3 committee for the 10BASE-T Draft 11 (P802.3I/D11) specification for a 10-Mb/s, CSMA/CD, twisted-pair Ethernet. The 82506TB is used as the interface between the attachment unit interface (AUI) signals and the twisted pair. Two strapping options are available.

- **Link Integrity Disable (LID).** When the LID strapping option is enabled (driven high), the link integrity function is disabled. When driven low, link pulses are transmitted on the twisted pair medium in the absence of data transmissions. In addition, the receiver expects to see link pulses in the absence of receive data. If no receive data or link pulses are received within 100 ms ±50 ms the 82506TB will enter a link fail state. When LID is floated an internal pull-up biases the signal high.
- **Data Terminal Equipment (DTE).** When the DTE strapping option is enabled (driven high) the SQE test sequence is transmitted to the DTE after every successful transmission on the twisted pair network.

The 82506TB simplifies network management and troubleshooting by providing four status indicator LED drivers that monitor traffic on a node and report transmit, receive, collision, and jabber conditions.

Figure 3 is an example of a typical system configuration.

Transmit Path (AUI to TP)

The transmit portion of this component transfers data from the AUI to the twisted-pair analog filters. It also loops back the data to the RCV pair.

- **AUI Receiver.** The TP MAU receives transmit data from the data terminal equipment on the transmit pins(TRMT/$\overline{\text{TRMT}}$) of the AUI-DO circuit (as defined by the IEEE802.3-1988 specification). The 82506TB then transmits the data onto the twisted-pair cable via the twisted-pair drivers. The AUI transmit inputs must be transformer coupled to the TRMT/$\overline{\text{TRMT}}$ pins. For best operation, the AUI signal should be dc biased to a common mode voltage of $V_{CC}/2$.

  The squelch circuit rejects (filters) all signals with an amplitude less than 160 mV or a pulse width less than 20 ns. A signal with an amplitude greater than 300 mV and a pulse width greater than 75 ns is accepted and turns off the squelch filter. The squelch filter remains off until an IDL pulse is detected or until the input does not exceed the detection threshold for 500 ± 100 ns.

- **AUI Receive Signal Levels.** The receiver (TRMT/$\overline{\text{TRMT}}$) is able to recognize differential signals as small as 300-mV peak. Internal circuitry samples the common mode voltage to provide full differential signal detection.
- **TP Driver Characteristics.** The drivers (HDAT/$\overline{\text{HDAT}}$ and LDAT/$\overline{\text{LDAT}}$) output CMOS logic levels with a source resistance less than 10 Ω and maximum current rating of 25 mA dc. All TP output driver pins are driven low as a result of any of the following.
  - Reception of an IDL signal.
  - A jabber condition is detected.
  - Activation of a link failure.
  - TRMT pair input fails to cross the detection threshold for 500 ± 100 ns.

When the driver detects the end of an IDL pulse on the TRMT pair, a timer of not more than 5 bit times (BT) is started. Activity on the TRMT pair is ignored until this timer expires.

Receive Path (TP to AUI)

When a RD signal is present, the receive circuit of the 82506TB transfers data from the RD pair input to the RCV pair output.

- **AUI Driver Characteristics.** This driver differentially drives a current onto the load connected between the RCV and $\overline{\text{RCV}}$ pins. The current through the load results in an output voltage between ±0.6 V and ±1.2 V measured differentially between the two pins. An external resistor (78 Ω) and capacitor (100 pF) must be connected for proper termination, as shown in Figure 4. This output is in accordance with the IEEE Spec 802.3 Sec.7.4.1 for MAUs. When the driver detects that it has finished sending an IDL pulse to the AUI it starts a timer of not more than 5 BT. Activity on the RD pair is ignored when this timer is functioning.
- **TP Receiver Threshold.** The TP receiver is connected to the output of a band limiting filter. The filter's input is transformer coupled to the twisted pair. The receiver is able to recognize differential signals as small as 350 mV peak. An external biasing circuit must provide a common mode voltage of $V_{CC}/2$. The differential input impedance of the RD pair is 20 kΩ ± 20%. Internal circuitry generates a dual-level bias voltage to determine proper signal level thresholds and prevent reception of spurious signals from the network (this is similar to a squelch function).

When the signal level at the RD input falls below −500 mV, with respect to the common mode voltage ± 10%, the data path is activated and the received signal is passed to the AUI cable. At the beginning of a reception the bias level at the RD input is reduced to −350 mV with respect to the common mode voltage ± 10%

Collision

The collision detection portion of the 82506TB senses the simultaneous presence of data on the TRMT and RD pins. It reacts by transmitting a 10-MHz square wave on the CLSN pair of the AUI cable. This signal is a periodic waveform of 10 MHz ± 15%, with a duty cycle no worse than 40/60 or 60/40. It is transmitted within 9 bit times after the component detects a collision (as specified by 10BASE-T Draft 11, Sec.14.2.1.3). If the receive pair becomes active while the transmit pair is active, the loopback data on the RCV pair switches from transmit data to receive data within 13 ± 3 BT from the assertion of the CLSN pair. If the RD pair goes active while the TRMT pair is active a collision condition will be detected and the SQE will continue for 7 ± 2 BT. If a collision condition exists where the TRMT pair has gone idle while the RD pair is still active, SQE can continue for up to 9 BT.

The collision AUI driver differentially drives a signal onto the load connected between the CLSN pair.

This driver differentially drives a current onto the load connected between the CLSN and $\overline{\text{CLSN}}$ pins. The current through the load results in an output voltage between ±0.6 V and ±1.2 V measured differentially between the two pins. An external resistor (78 Ω) and capacitor (100 pF) must be connected for proper termination, as shown in Figure 4. The output is in accordance with IEEE 802.3-1988 Sec. 7.4.1 for the AUI.

Jabber (Watchdog Timer)

The 82506TB supports a self-interrupt function that protects the network from a jabbering node (i.e., continuous transmission). The component provides a nominal window of 50 ms during the time a normal data link frame can be transmitted. If the frame length exceeds this duration, the component immediately inhibits all further transmission of that frame and activates the CLSN pair (as specified by IEEE 802.3-1988 Sec. 8.2.1.5). When activity on the TRMT pair has ceased, the component continues to present the CSO signal to the CLSN pair for 0.5 s ± 50%. The component then resets itself and returns to the idle state (as specified by the 10BASE-T Draft 11 14.2.1.5). The transmission of link integrity pulses from the TP drivers is not inhibited when the TP MAU jabber is activated and link integrity is enabled.

SQE Test (Heartbeat)

The SQE test begins within 11 ± 5 bit times after the TRMT pair detects the IDL signal. The SQE test duration is 10 ± 5 bit times. When the AUI-DO circuit has gone idle after a successful transmission (without a collision), and the DTE input is high, the 82506TB activates the CLSN pair to simulate a collision.

Link Integrity

The link integrity function determines if the receive twisted-pair link is faulty. Enabling the function (LID tied to V_{SS}) causes the RLED receive traffic indicator to display the status of the receive twisted-pair link. The link integrity function permits the active disabling of the transmit and loopback paths within the TP MAU component in response to a link integrity fault. The link integrity function monitors the RD pair for either data or link test pulses by providing a 100 ms ± 50 ms window during which data or a link test pulse is expected. If this timer expires and the LID is off, the RLED indicator is turned off and the component's transmit and loopback capabilities are disabled. The 82506TB remains in a link fail state until after a data packet is received, or until after a sequence of consecutive link test pulses are received. The sequence length is between two and ten pulses. If a pulse or receive traffic is detected within this window, the timer is reset and the RLED indicator remains on.

The TP MAU also transmits link test pulses onto the transmit twisted pair link when link integrity is enabled. In the absence of transmit traffic, a link test pulse is transmitted at a nominal rate of one pulse each 16 ms ± 8 ms. If the link integrity is disabled, the RLED indicator remains on in the absence of receive traffic, (data and link pulses). Received link test pulses are also ignored at the RD pair input.

LED Status

Four light-emitting diodes (LEDs) give the user a visual indication of the MAU's status. The 82506TB provides the logic signals needed to drive the LEDs.

- **XLED.** The following LED values (on or off) are used to indicate transmission (AUI) status.
  - The LED is normally on, which indicates no transmission is in progress.
  - The LED is off when a valid packet is transmitted. The duration of the off period is 100 ms ± 10 ms. The minimum duration of the on period is 4 ms while waiting for next valid packet transmission.

- **RLED**—With LID Disabled. The following LED values (on or off) are used to indicate reception (TP) status.
  - The LED is normally on, which indicates there is no receive traffic.
  - The LED is off when a valid packet is received. The duration of the off period is 100 ms ± 10 ms. The minimum duration of the on period is 4 ms while waiting for next valid packet receive.
- **RLED**—With LID Enabled. The following LED values (on or off) are used to indicate reception (TP) status.
  - The LED is normally on, which indicates no receive traffic and successful reception of the link test pulse.
  - The LED turns off if no receive traffic or link integrity signals have been received for more than 0.5 s. This visually indicates a failure of a link segment. The LED remains off until a link test pulse, or receive traffic, is successfully detected, after which the LED is turned on with a minimum on time of 0.5 s.
  - If the link is working, the LED will be turned off when a valid data packet is received from the twisted pair. When a packet is received, the LED is turned off for a duration of 100 ms ± 10 ms, then the LED is turned back on.
  - The LED remains on for a minimum of 4 ms; it is turned off when the next packet is received.
- **CLED.** The following LED values indicate collision status.
  - The LED is normally off, which indicates no collision.
  - The LED is turned on when a collision is detected. It remains on for a nominal time of 15 ms ± 5 ms, after which it is turned off.
  - The LED may be turned back on immediately upon detection of another collision. There is no minimum off time.
  - If a collision occurs while the LED is on, the LED remains on for the nominal time following the last detected collision.
- **JLED.** The following LED values indicate the jabber status.
  - The LED is normally off, which indicates a no jabber condition.
  - The LED is turned on when the watchdog timer times out, and the TP drivers are disabled. It remains on until the jabber condition is corrected.
  - The LED is turned off after the watchdog timer counts out the 0.5 s ± 0.25 s reset time.

LED Drivers

The typical LED circuit consists of an external resistor in series with the LED and connected the V_{CC}. The LED driver pulls the pin low to turn the LED on. Each LED driver can sink up to 15 mA of current, with an output impedance of less than 50 Ω.

Clock Generation

A 20-MHz, parallel-resonant crystal is used to control the clock generation oscillator of the TP MAU. We recommend that the crystal meet the following specifications.

- Quartz crystal
- 20 MHz ± 0.01%
- Parallel resonant with a 20-pF load fundamental mode with a maximum series resistance of 25 Ω.

The crystal shunt and external capacitance should be less than 10 pF. The crystal should be connected adjacent to the 82506TB to the XTAL and V_{SS} pins. The crystal shunt capacitance (CO) should not exceed 5 pF.

An external MOS-level clock can be applied to the crystal oscillator input. A resistor should be added in series with the clock source to limit the amplitude of the voltage swing seen by the pin. A 500-Ω resistor works well in most cases. If users are concerned about the duty-cycle variation caused by driving the TPMAU with a clock source, the following test can be done on the bench to empirically determine the best resistor value for the user's application.

- Place the part in dc test mode, as described in the Test Mode section of this document.
- Attach an oscilloscope to the JLED pin. This pin outputs the internal clock source.
- Alter the resistor value to obtain an optimal duty-cycle ratio. Experiments have shown that a 500-Ω resistor works well for LS TTL logic levels; CMOS logic levels need a 1-kΩ resistor.

Under no circumstances should the clock be driven straight into the TPMAU. Also, under no circumstances should the clock stop, not even briefly, once power is applied to the TPMAU. If the clock to the TPMAU is stopped, power to the TPMAU must be removed to ensure proper behavior of the TPMAU.

Strapping Options

All strapping options are connected to internal pull-up resistors, (nominally 100 kΩ). A resistor tying a strapping option low must be able to sink 70 μA.

Test Mode

The 82506TB enters the ac or dc test mode when the test pin (TEST) is held high. The ac test mode is activated by also holding the DTE pin high; the internal clock speeds are increased by three to reduce

the ac test time. The dc test is activated by holding the DTE pin low while TEST is held high. During the dc test the oscillator frequency and duty cycle can be tested on the JLED pin and the three internal clocks (ACK, BCK, and CCK) can be tested on the XLED, RLED,and CLED pins respectively. The AUI driver current can be measured with a 39-Ω resistor between the receive pair pins.

Power Considerations

There are seven power connections to the TP MAU—three pairs of V_{CC} and V_{SS} connections and a fourth V_{SS} pin for the XTAL oscillator. Table 2 describes which internal circuits are powered by each V_{CC}/V_{SS} pair.

Table 2. INTERNAL CIRCUIT

Pin No.	Internal Circuits
1, 28	**Analog Supplies.** Analog signal receivers, energy detection circuits, delay lock loop, and band gap reference.
10, 11	**AUI Output Drivers.** Digital polycells, XTAL oscillator (V_{CC} only), and LED drivers (V_{CC} only).
14	GND only for XTAL oscillator and pins 15, 16, and 17.
18, 23	TP CMOS output drivers only.

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Ambient Operating Temperature (TA) 0 to +70°C

Storage Temperature −40 to +125°C

All Output and Supply Voltages −0.5 V to V_{CC} +0.5V

All Input Voltages −0.5 V to V_{CC} +0.5V

NOTICE: This data sheet contains preliminary information on new products in production. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

DC Characteristics T_A = 0 to 70°C, V_{CC} = 5 V ± 5%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{AID}	Input Differential Voltage (AUI)	0.300	1.3	V	
V_{TID}	Input Differential Voltage (TP)	0.350	2.0	V	
V_{XTL}	XTAL Input Low Voltage		0.4	V	
V_{XCH}	XTAL Input High Voltage	3.9		V	
V_{IL}	Input Low Voltage		0.2	V	
V_{IH}	Input High Voltage	4.0		V	
V_{AOD}	Output Differential Voltae (AUI)	0.600	1.2	V	
V_{TOL}	Output Voltage Low (TP)		0.1	V	V_{CC} = 5.0 V, R_L = 500 Ω
V_{TOH}	Output Voltage High (TP)	4.9		V	V_{CC} = 5.0 V, R_L = 500 Ω
V_{OL}	Output Voltage Low		0.13	V	V_{CC} = 5.0 V, R_L = 2000 Ω
V_{OH}	Output Voltage High	4.87		V	V_{CC} = 5.0 V, R_L = 2000 Ω
R_S	TP Driver Series Impedance		10	Ω	V_{CC} = 4.5 V, I = 25 mA (max)
I_{CC}	Power Supply Current with a Traffic Load		145	mA	V_{CC} = 5.00 V
PD	*Power Dissipation* with a Traffic Load		0.6	W	V_{CC} = 5.0 V

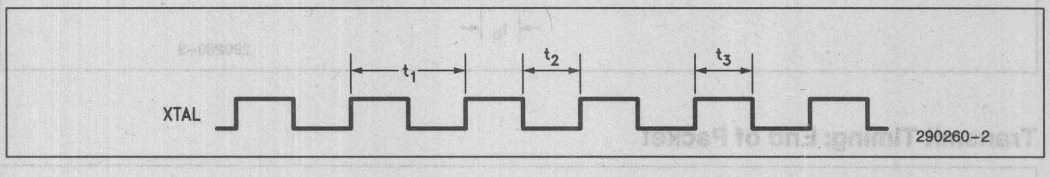
AC Timing Conditions

1. T_A = 0 to 70°C, V_{CC} = 5 V ± 5%.
2. Timing measurement points are 50% points unless otherwise noted.

Clock Timing

Symbol	Parameter	Min	Max	Units
t_1	XTAL (or Oscillator) Frequency	18	22	MHz
t_2	XTAL High and Low Times	22.5	27.5	ns

Clock Timing

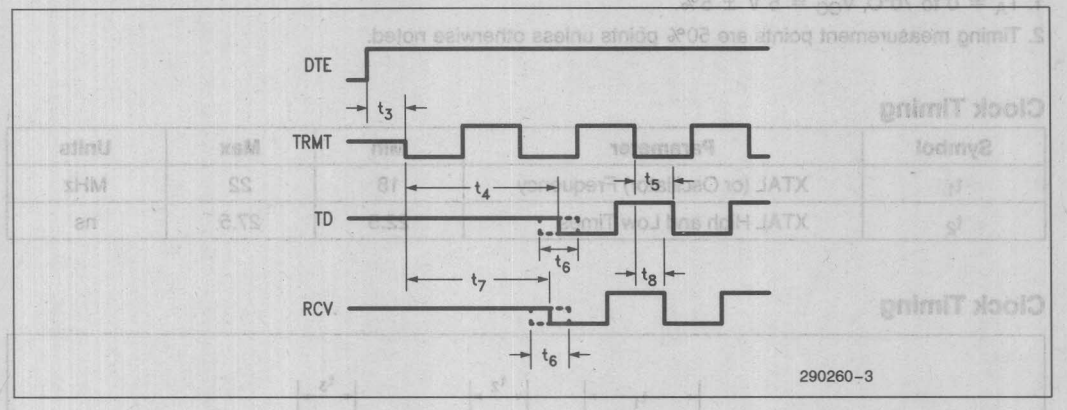


Transmit Timing

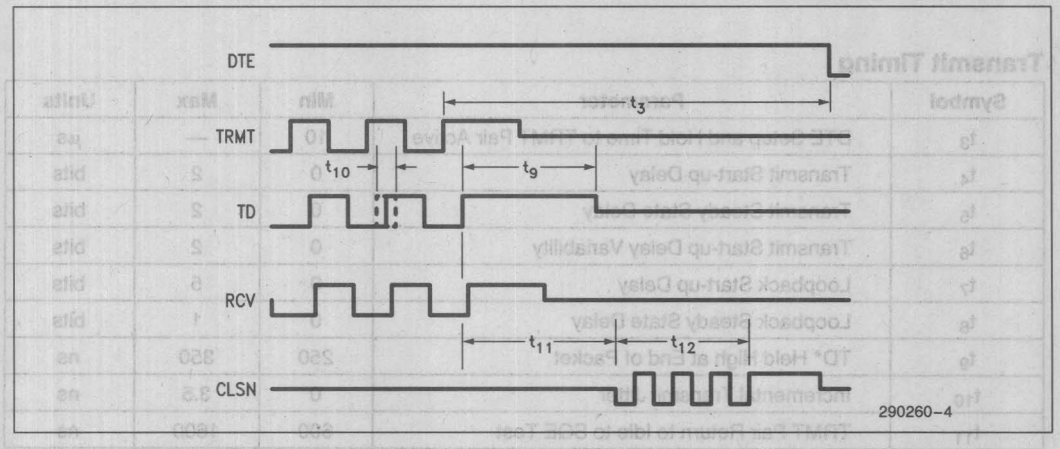
Symbol	Parameter	Min	Max	Units
t_3	DTE Setup and Hold Time to TRMT Pair Active	10	—	μs
t_4	Transmit Start-up Delay	0	2	bits
t_5	Transmit Steady State Delay	0	2	bits
t_6	Transmit Start-up Delay Variability	0	2	bits
t_7	Loopback Start-up Delay	0	5	bits
t_8	Loopback Steady State Delay	0	1	bits
t_9	TD* Held High at End of Packet	250	350	ns
t_{10}	Incremental Transmit Jitter	0	3.5	ns
t_{11}	TRMT Pair Return to Idle to SQE Test	600	1600	ns
t_{12}	SQE Test Duration	500	1500	ns

*TD represents the differential voltage between the signals HDAT and $\overline{\text{HDAT}}$

Transmit Timing: Start of Packet



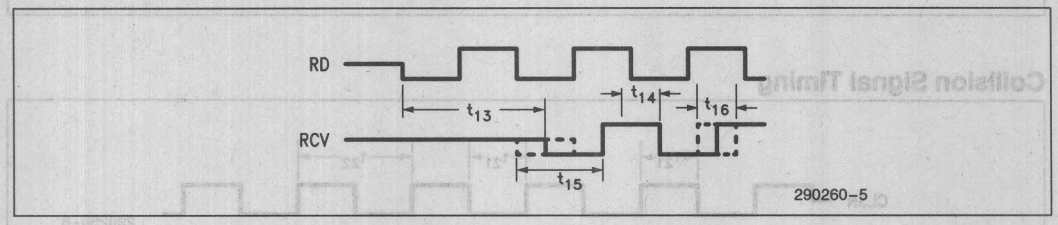
Transmit Timing: End of Packet



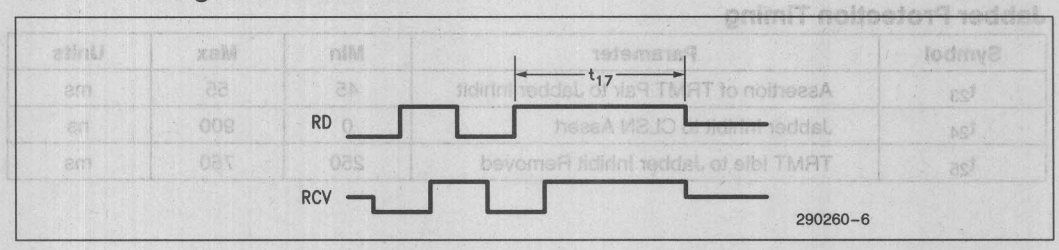
Receive Timing

Symbol	Parameter	Min	Max	Units
t_{13}	Receive Start-up Delay	0	5	bits
t_{14}	Receive Steady State Delay	0	2	bits
t_{15}	Receive Start-up Delay Variability	0	2	bits
t_{16}	Incremental Receive Jitter	0	1.5	ns
t_{17}	RD Pair Held High at End of Packet	0	300	ns

Receive Timing: Start of Packet



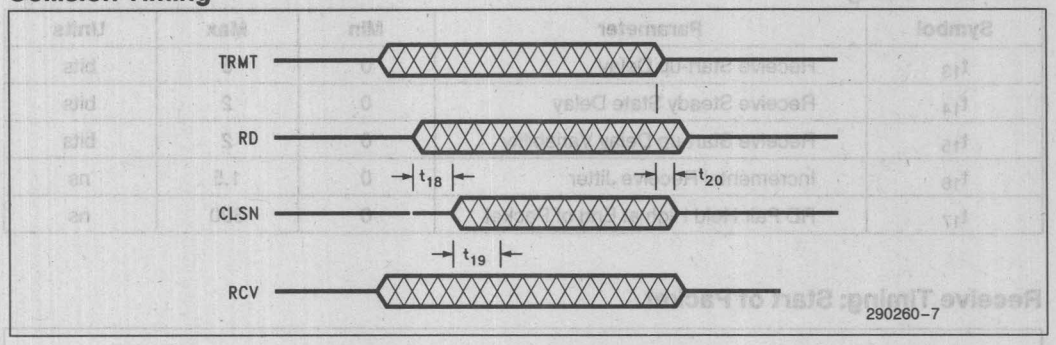
Receive Timing: End of Packet



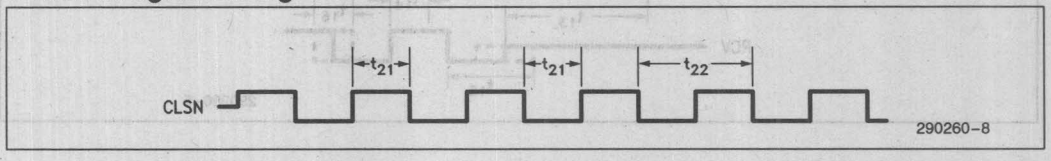
Collision Timing

Symbol	Parameter	Min	Max	Units
t_{18}	Onset of Collision to CLSN Pair Assertion	0	900	ns
t_{19}	RCV Pair Source to RD after CLSN Assert	0	900	ns
t_{20}	End of Collision to CLSN Pair Return to Idle	0	900	ns
t_{21}	CLSN Pair High/Low Time	40	60	ns
t_{22}	CLSN Pair Frequency	8.5	11.5	MHz

Collision Timing



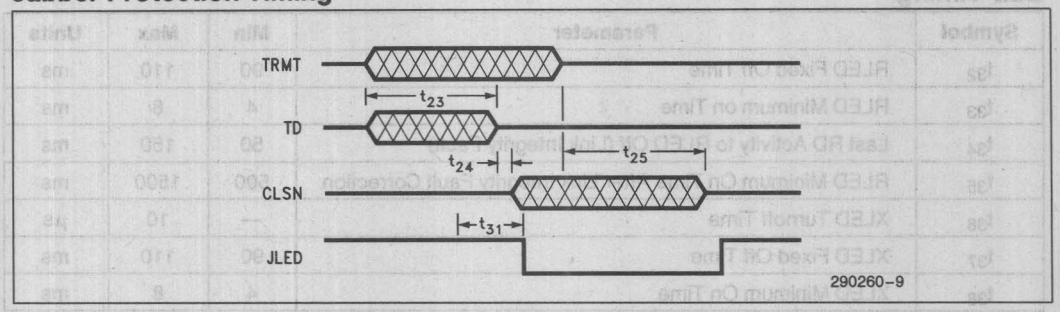
Collision Signal Timing



Jabber Protection Timing

Symbol	Parameter	Min	Max	Units
t_{23}	Assertion of TRMT Pair to Jabber Inhibit	45	55	ms
t_{24}	Jabber Inhibit to CLSN Assert	0	900	ns
t_{25}	TRMT Idle to Jabber Inhibit Removed	250	750	ms

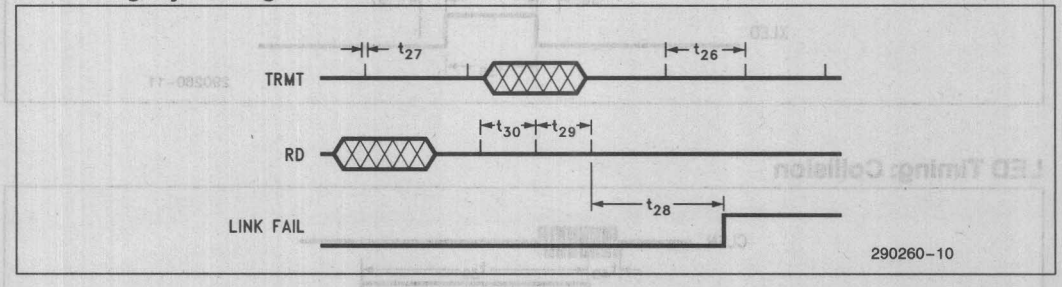
Jabber Protection Timing



Link Integrity Timing

Symbol	Parameter	Min	Max	Units
t_{26}	Last TRMT Pair Activity to Link Test Pulse	8	24	ms
t_{27}	Link Test Pulse Width	80	120	ns
t_{28}	Last RD Activity to Link Integrity Fault Assert	50	150	ms
t_{29}	Minimum Idle Time Between Consecutive Leakbeat Reception	6	8	ms
t_{30}	Maximum Idle Time Between Consecutive Leakbeat Reception	24	150	ms
t_{31}	JLED Turn-on Time	—	10	μs

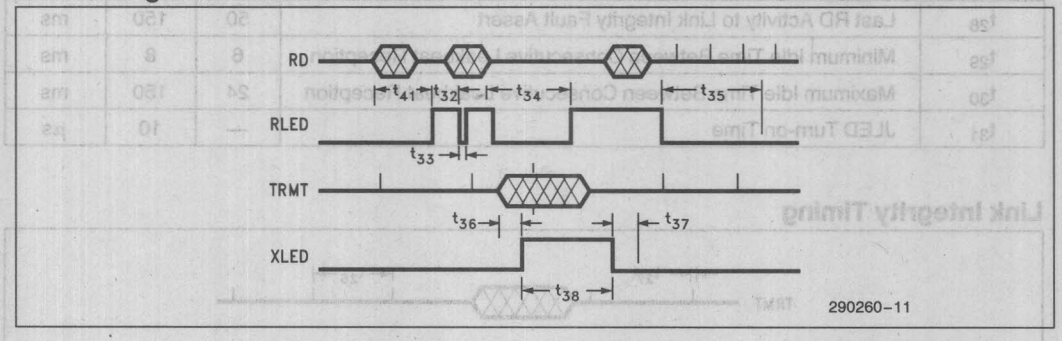
Link Integrity Timing



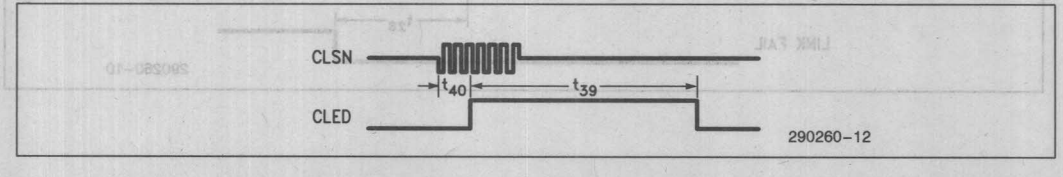
LED Timing

Symbol	Parameter	Min	Max	Units
t_{32}	RLED Fixed Off Time	90	110	ms
t_{33}	RLED Minimum on Time	4	8	ms
t_{34}	Last RD Activity to RLED Off (Link Integrity Fault)	50	150	ms
t_{35}	RLED Minimum On Time After Link Integrity Fault Correction	500	1500	ms
t_{36}	XLED Turnoff Time	—	10	μs
t_{37}	XLED Fixed Off Time	90	110	ms
t_{38}	XLED Minimum On Time	4	8	ms
t_{39}	CLED Turnoff Time	—	10	μs
t_{40}	CLED Nominal On Time	10	20	ms
t_{41}	RLED Turnoff Time	—	10	μs

LED Timing: Transmit and Receive



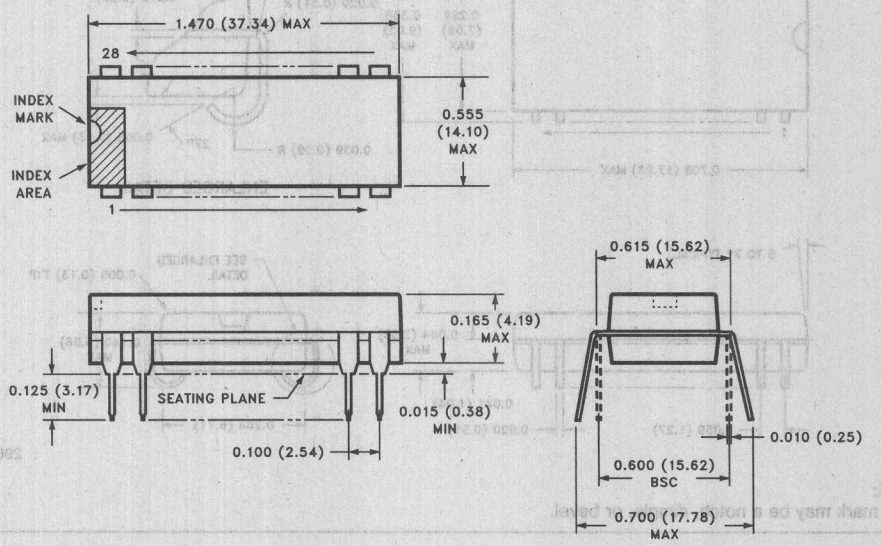
LED Timing: Collision



Outline Diagrams

Dimensions are in inches and (millimeters).

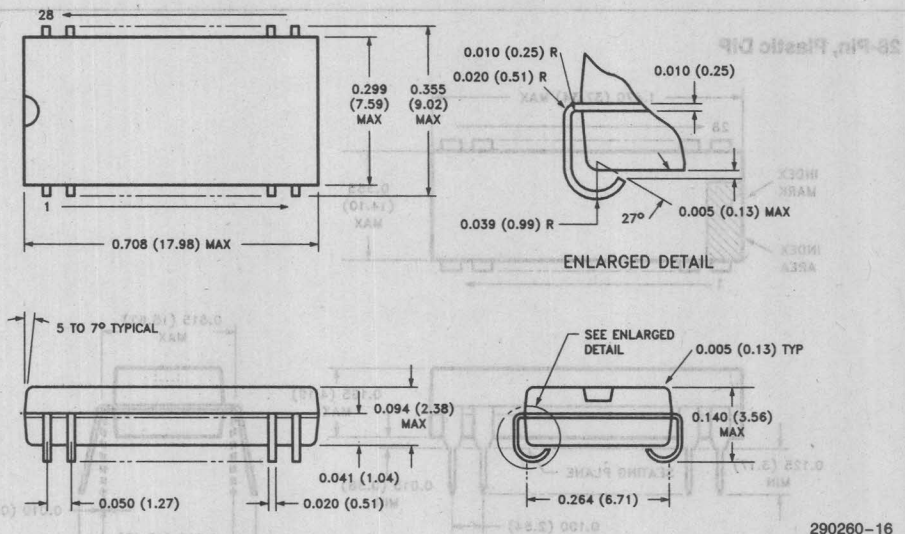
28-Pin, Plastic DIP



290260-15

NOTES:
Meets JEDEC standards.
Index mark may be a semicircular notch or circular dimple located in the index area.

28-Pin, Plastic SOJ
Dimensions are in inches and (millimeters)



NOTE:
Index mark may be a notch, dimple, or bevel.

1

October 1990

An 82586 Data Link Driver

CHARLES YAGER

Order Number: 231421-002

AN 82586 DATA LINK DRIVER

CONTENTS

PAGE

INTRODUCTION 1-339

1.0 FITTING THE SOFTWARE INTO THE OSI MODEL 1-339

2.0 LARGE MODEL COMPILATION ... 1-340

3.0 THE 82586 HANDLER 1-340
3.1 The Buffer Model 1-340
3.2 The Handler Interface 1-341
3.3 Initialization 1-343
3.3.1 Building the CB and RFA Pools ... 1-343
3.3.2 82586 Initialization 1-343
3.3.3 Self Test Diagnostics 1-344
3.4 Command Processing 1-345
3.4.1 Accessing Command Blocks 1-345
3.4.2 Issuing CU Commands 1-345
3.4.3 Interrupt Service Routine 1-346
3.4.4 Sending Frames 1-346
3.4.5 Accessing Transmit Buffers 1-347
3.4.6 Multicast Addresses 1-347
3.4.7 Resetting the 82586 1-348
3.5 Receive Frame Processing 1-349
3.5.1 Receive Interrupt Processing 1-349
3.5.2 Returning FDs and RBDs 1-349
3.5.3 Restarting the Receive Unit 1-349

4.0 LOGICAL LINK CONTROL 1-350
4.1 Adding and Deleting LSAPs 1-352

5.0 APPLICATION LAYER 1-352
5.1 Application Layer Human Interface .. 1-352
5.2 A Sample Session 1-353
5.3 Terminal Mode 1-355
5.3.1 Sending Frames 1-356
5.3.2 Receiving Frames 1-356
5.4 Monitor Mode 1-356
5.5 High Speed Transmit Mode 1-357

APPENDIX A: COMPILING, LINKING, LOCATING, AND RUNNING THE SOFTWARE ON THE iSBC 186/51 BOARD 1-358

INTRODUCTION

This application note describes a design example of an IEEE 802.2/802.3 compatible Data Link Driver using the 82586 LAN Coprocessor. The design example is based on the "Design Model" illustrated in "Programming the 82586". It is recommended that before reading this application note, the reader clearly understands the 82586 data structures and the Design Model given in "Programming the 82586".

"Programming the 82586" discusses two basic issues in the design of the 82586 data link driver. The first is how the 82586 handler fits into the operating system. One approach is that the 82586 handler is treated as a "special kind of interface" rather than a standard I/O interface. The special interface means a special driver that has the advantage of utilizing the 82586 features to enhance performance. However the performance enhancement is at the expense of device dependent upper layer software which precludes the use of a standard I/O interface.

The second issue "Programming the 82586" discusses is which algorithms to choose for the CPU to control the 82586. The algorithms used in this data link design are taken directly from "Programming the 82586". Command processing uses a linear static list, while receive processing uses a linear dynamic list.

The application example is written in C and uses the Intel C compiler. The target hardware for the Data Link Driver is the iSBC 186/51 COMMputer, however a version of the software is also available to run on the LANHIB Demo board.

1.0 FITTING THE SOFTWARE INTO THE OSI MODEL

The application example consists of four software modules:

- Data Link Driver (DLD): drives the 82586, also known as the 82586 Handler.
- Logical Link Control (LLC): implements the IEEE 802.2 standard.
- User Application (UAP): exercises the other software modules and runs a specific application.
- C hardware support: written in assembly language, supports the Intel C compiler for I/O, interrupts, and run time initialization for target hardware.

Figure 1 illustrates how these software modules combined with the 82586, 82501 and 82502 complete the first two layers of the OSI model. The 82502 implements an IEEE 802.3 compatible transceiver, while the 82501 completes the Physical layer by performing the serial interface encode/decode function.

The Data Link Layer, as defined in the IEEE 802 standard documents, is divided into two sublayers: the Logical Link Control (LLC) and the Medium Access Control (MAC) sublayers. The Medium Access Control sublayer is further divided into the 82586 Coprocessor plus the 82586 Handler. On top of the MAC is the LLC software module which provides IEEE 802.2 compatibility. The LLC software module implements the Station Component responses, dynamic addition and deletion of Service Access Points (SAPs), and a class 1 level of service. (For more information on the LLC sublayer, refer to IEEE 802.2 Logical Link Control Draft Standard.) The class 1 level of service provides a connectionless datagram interface as opposed to the class 2 level of service which provides a connection oriented level of service similar to HDLC Asynchronous Balanced Mode.

On top of the Data Link Layer is the Upper Layer Communications Software (ULCS). This contains the Network, Transport, Session, and Presentation Layers. These layers are not included in the design example, therefore the application layer of this ap note interfaces directly to the Data Link layer.

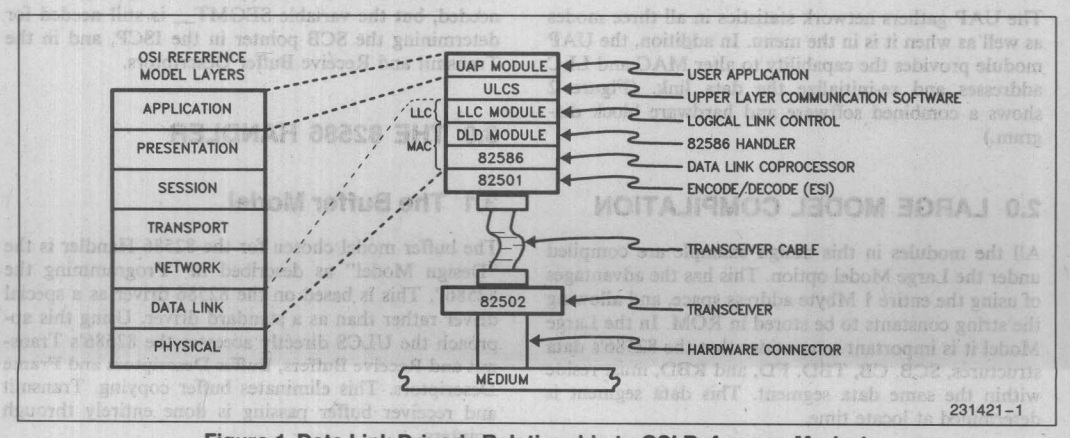


Figure 1. Data Link Driver's Relationship to OSI Reference Mode 1

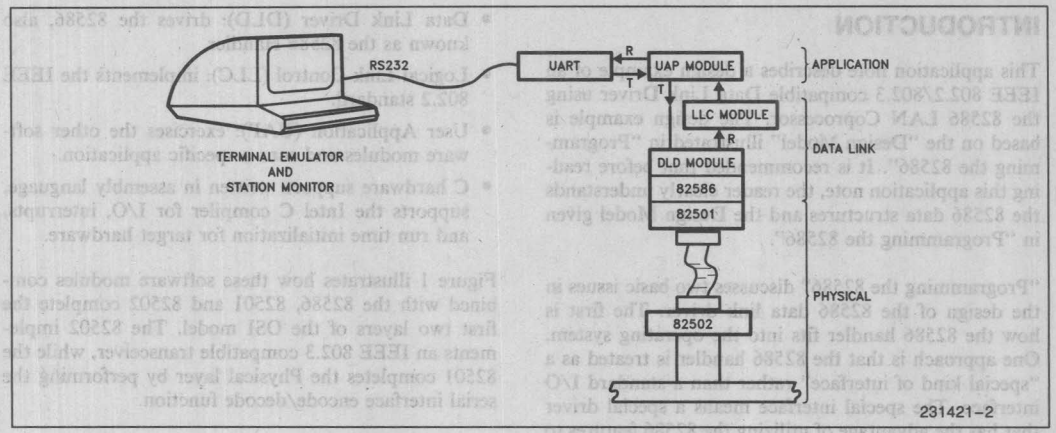


Figure 2. Block Diagram of the Hardware and Software

The application layer is implemented in the User Application (UAP) software module. The UAP module operates in one of three modes: Terminal Mode, Monitor Mode, and High Speed Transmit Mode. The software initially enters a menu driven interface which allows the program to modify several network parameters or enter one of the three modes.

The Terminal Mode implements a virtual terminal with datagram capability (connectionless "class 1" service). This mode can also be thought of as an async to IEEE 802.3/802.2 protocol converter.

The Monitor Mode provides a dynamic update on the terminal of 6 station related parameters. While in the monitor mode, any size frame can be repeatedly transmitted to the cable in a software loop.

High Speed Transmit Mode transmits frames to the cable as fast as the software possibly can. This mode demonstrates the throughput performance of the Data Link Driver.

The UAP gathers network statistics in all three modes as well as when it is in the menu. In addition, the UAP module provides the capability to alter MAC and LLC addresses and re-initialize the data link. (Figure 2 shows a combined software and hardware block diagram.)

2.0 LARGE MODEL COMPILATION

All the modules in this design example are compiled under the Large Model option. This has the advantages of using the entire 1 Mbyte address space, and allowing the string constants to be stored in ROM. In the Large Model it is important to consider that the 82586's data structures, SCB, CB, TBD, FD, and RBD, must reside within the same data segment. This data segment is determined at locate time.

The C__Assy__Support module has a run time start off function which loads the DLD data segment into a global variable SEGMT__. This data segment is used by the 82586 Handler for address translation purposes. The 82586 uses a flat address while the 80186 uses a segmented address. Any time a conversion between 82586 and 80186 addresses are needed the SEGMT__ variable is used.

Pointers for the 80186 in the large model are 32 bits, segment and offset. All the 82586 link pointers are 16 bit offsets. Therefore when trading pointers between the 82586 and the 80186, two functions are called: Offset (ptr), and Build__Ptr (offset). Offset (ptr) takes a 32 bit 80186 pointer and returns just the offset portion for the 82586 link pointer. While Build__Ptr (offset) takes an 82586 link pointer and returns a 32 bit 80186 pointer, with the segment part being the SEGMT__ variable. Offset ( ) and Build__Ptr( ) are simple functions written in assembly language included in the C__Assy__Support module.

In the small model, Offset ( ) and Build__Ptr( ) are not needed, but the variable SEGMT__ is still needed for determining the SCB pointer in the ISCP, and in the Transmit and Receive Buffer Descriptors.

3.0 THE 82586 HANDLER

3.1 The Buffer Model

The buffer model chosen for the 82586 Handler is the "Design Model" as described in "Programming the 82586". This is based on the 82586 driver as a special driver rather than as a standard driver. Using this approach the ULCS directly accesses the 82586's Transmit and Receive Buffers, Buffer Descriptors and Frame Descriptors. This eliminates buffer copying. Transmit and receiver buffer passing is done entirely through pointers.

The only hardware dependencies between the Data Link and ULCS interface are the buffer structures. The ULCS does not handle the 82586's CBs, SCB or initialization structures. To isolate the data link interface from any hardware dependencies while still using the design model, another level of buffer copying must be introduced. For example, when the ULCS transmits a frame it would have to pass its own buffers to the data link. The data link then copies the data from ULCS buffers into 82586 buffers. When a frame is received, the data link copies the data from the 82586's buffers into the ULCS buffers. The more copying that is done the slower the throughput. However, this may be the only way to fit the data link into the operating system. The 82586 Handler can be made hardware independent by adding a receive and transmit function to perform the buffer copying.

The 82586 Handler allocates buffers from two pools of memory: the Transmit pool, and the Receive pool as illustrated in Figure 3. The Transmit pool contains Transmit Buffer Descriptors (TBDs) and Transmit Buffers (TBs). The Receive pool contains Frame Descriptors (FDs), Receive Buffer Descriptors (RBDs), and Receive Buffers (RBs).

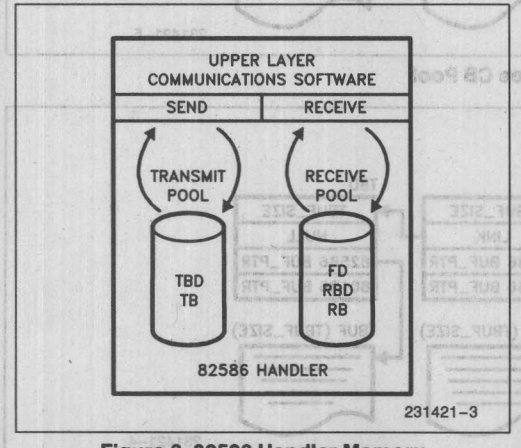


Figure 3. 82586 Handler Memory Management Model

When the ULCS wants to transmit, it requests a TBD from the handler. The handler returns a pointer to a free TBD. Each TBD has a TB attached to it. The ULCS fills the buffer, sets the appropriate fields in the TBD, and passes the TBD pointer back to the handler for transmission. After the frame is transmitted, the handler places the TBD back into the free TBD pool. If the ULCS needs more than one buffer per frame, it simply requests another TBD from the handler and performs the necessary linkage to the previous TBD.

On the receive side, the RFA pool is managed by the 82586 itself. When a frame is received, the 82586 interrupts the handler. The handler passes a FD pointer to the ULCS. Linked to the FD is one or more RBDs and RBs. The ULCS extracts what it needs from the FD, RBDs and RBs, and returns the FD pointer back to the handler. The handler places the FD and RBDs back into the free RFA pool.

3.2 The Handler Interface

The handler interface provides the following basic functions:

- initialization
- sending and receiving frames
- adding and deleting multicast addresses
- getting transmit buffers
- returning receive buffers

Figure 4 lists the Handler Interface functions.

On power up, the initialization function is called. This function initializes the 82586, and performs diagnostics. After initialization, the handler is ready to transmit and receive frames, and add and delete multicast addresses.

To send a frame, the ULCS gets one or more transmit buffers from the handler, fills them with data, and calls the send function. When a frame is received, the handler calls a receive function in the ULCS. The ULCS receive function removes the information it needs and returns the receive buffers to the handler. The addition and deletion of multicast addresses can be done "on the fly" any time after initialization. The **receiver doesn't have to be disabled** when this is done.

The command interface to the handler is totally asynchronous—the ULCS can issue transmit commands or multicast address commands whenever it wants. The commands are queued by the handler for the 82586 to execute. If the command queue is full, the send frame procedure returns a false status rather than true. The size of the command queue can be set at compile time by setting the CB—CNT constant. Typically the command queue never has more than a few commands on it because the 82586 can execute commands faster than the ULCS can issue them. This is not the case in a heavily loaded network when deferrals, collisions, and retries occur.

The command interface to the 82586 handler is hardware independent; the only hardware dependence is the buffering. A hardware independent command interface doesn't have any performance penalty, but some 82586 programmability is lost. This shouldn't be of concern since most data links do not change configuration parameters during operation. One can simply modify a few constants and recompile to change frame and network parameters to support other data links.

Figure 4. List of Handler Interface Functions

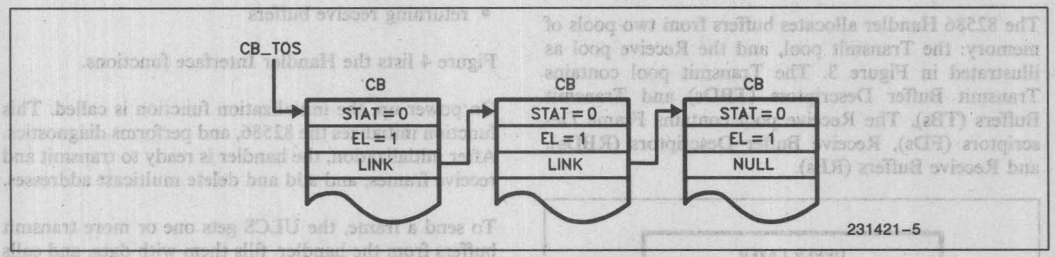


Figure 5. Free CB Pool

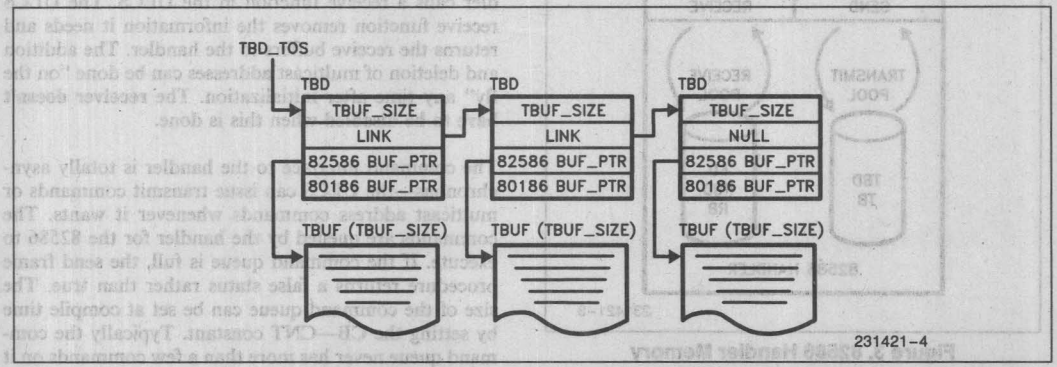


Figure 6. Free Transmit Buffer Descriptor Pool

Phase 1 executes initialization of all the handlers flags, interrupt vectors, counters, and 82586 data structures. There are two separate functions which initialize the CB and RFA pools: Build__CB( ) and Build__Rfa( ).

3.3.1 BUILDING THE CB AND RFA POOLS

Build__CB( ) builds a stack of free linked Command Blocks, and another stack of free linked Transmit Buffer Descriptors. (See Figures 5 and 6.) Each stack has a Top of Stack pointer, which points to the next free structure. The last structure on the list has a NULL link pointer.

the RBD list. The RBDs are initialized with both 82586 and 80186 buffer pointers. The 80186 buffer pointer is added to the end of the RBD structure. Begin and end pointers are used to mark the boundaries of the free lists.

3.3.2 82586 INITIALIZATION

The 82586 initialization data structure SCP is already set since it resides in ROM, however, the ISCP must be loaded with information. Within the SCP ROM is the pointer to the ISCP; the ISCP is the only absolute address needed in the software. Once the ISCP address is determined, the ISCP can be loaded. The SCB base is obtained from the C__Assy__Support module. The global variable SEGMT__contains the address of the

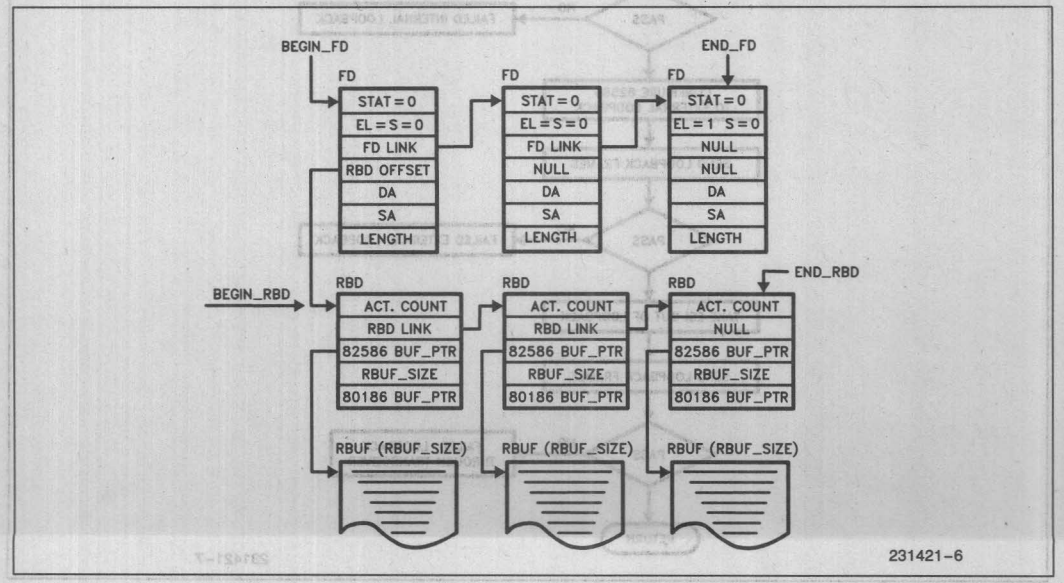


Figure 7. Free RFA

data segment of the handler. The 80186 shifts this value to the left by 4 and loads it into the SCB base. The SCB offset is now determined by taking the 32 bit SCB pointer and passing it to the Offset( ) function.

The 82586 interrupt is disabled during initialization because the interrupt function is not designed to handle 82586 reset interrupts. To determime when the 82586 is finished with its reset/initialization, the SCB status is polled for both the CX and CNA bits to be set. After the 82586 is initialized, both the CX and CNA interrupts are acknowledged.

The 82586 is now ready to execute commands. The Configuration is executed first to place the 82586 in internal loopback mode, followed by the IA command. The address for the IA command is read off of a prom on the PC board.

3.3.3 SELF TEST DIAGNOSTICS

The final phase of the handler initialization is to run the self test diagnostics. Four tests are executed: Diagnose command, Internal loopback, External loopback through the 82501, and External loopback through the transceiver. If these four tests pass, the data link is ready to go on line.

The function that executes these diagnostics is called Test__Link( ). If any of the tests fail, Test__Link( ) returns immediately with the Self__Test global variable set to the type of failure. This Self__Test global variable is then returned to the function which originally called Init__586( ). Therefore Init__586( ) can return one of five results: FAILED__DIAGNOSE, FAILED__LPBK__INTERNAL, FAILED__LPBK__EXTERNAL, FAILED__LPBK__TRANSCEIVER or PASSED.

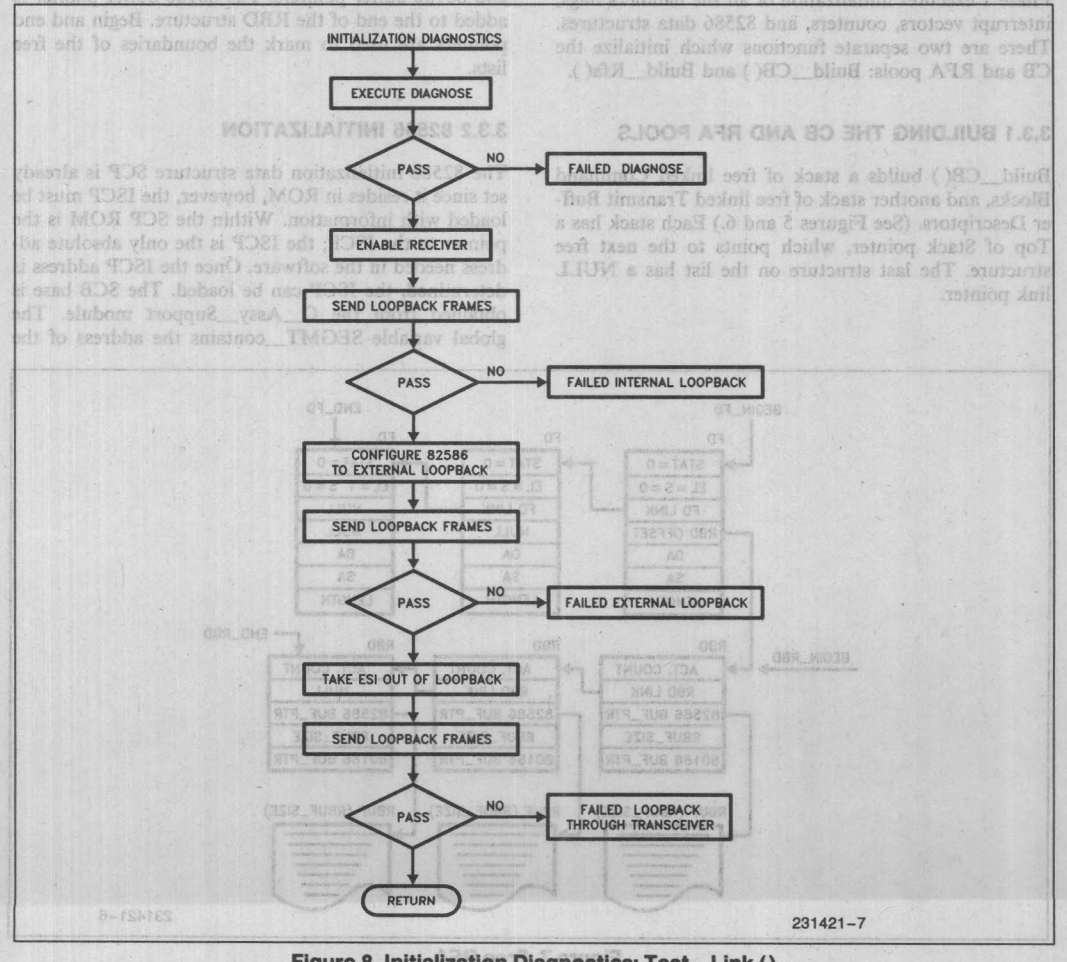


Figure 8. Initialization Diagnostics: Test__Link ()

The Diagnose( ) function, called by Test_Link( ), does not return until the diagnose command is completed. If the interrupt service routine detects that a Diagnose command was completed then it sets a flag to allow the Diagnose( ) function to return, and it also sets the Self_Test variable to FAIL if the Diagnose command failed. If the Diagnose command completed successfully, the loopback tests are performed.

Before any loopback tests are executed, the Receive Unit is enabled by calling Ru_Start( ). Loopback tests begin by calling Send_Lpbk_Frame( ), which sends 8 frames with known loopback data and its own destination address. More than one loopback frame is sent in case one or more of them are lost. Also several of the frames will have been received by the time flags.lpbk_test is checked.

Two flag bits are used for the loopback tests: flags.lpbk_mode, and flags.lpbk_test. flags.lpbk_mode is used to indicate to the receive section that the frames received are potentially loopback frames. The receive section will pass receive frames to the Loopback Check( ) function if the flags.lpbk_mode bit is set. The Loopback_Check( ) function first compares the source address of the frame with its station address. If this matches then the data is checked with the known loopback data. If the data matches, then the flags.lpbk_test bit is set, indicating a successful loopback. The flow of the Test_Link( ) function is displayed in Figure 8.

3.4 Command Processing

Command blocks are queued up on a static list for the 82586 to execute. The flow of a command block is given in Figure 9. When the handler executes a command it first has to get a free command block. It does this by calling Get_CB( ) which returns a pointer to a free command block. The CB structure is a generic one in which all commands except the MC-Setup can fit in. The handler then loads into the CB structure the type of command and associated parameters. To issue the command to the 82586 the Issue_CU_Cmd( ) function is called with the pointer to the CB passed to this function. Issue_CU_Cmd( ) places the command on the 82586's static command block list. After the 82586 executes the command, it generates an interrupt. The interrupt routine, Isr_586( ), processes the command and returns the Command Block to the free command block list by calling Put_Cb( ).

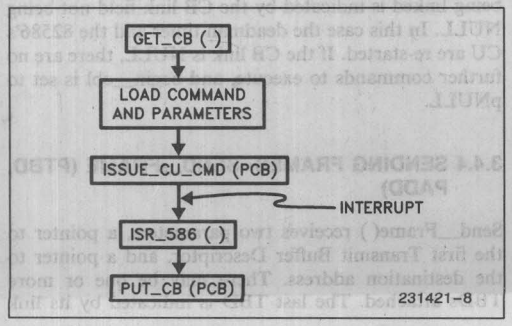


Figure 9. The Flow of a Command Block

3.4.1 ACCESSING COMMAND BLOCKS–GET_CB() and PUT_CB()

Get_Cb( ) returns a pointer to a free command block. The free command blocks are in a linear linked list structure which is treated as a stack. The pointer cb_tos points to the next available CB. Each time a CB is requested, Get_Cb( ) pops a CB off the stack. It does this by returning the pointer of cb_tos. cb_tos is then updated with the CB's link pointer. When the CB list is empty, Get_Cb( ) returns NULL.

There are two types of nulls, the 82586 'NULL' is a 16 bit offset, OFFFFH, in the 82586 data structures. The 80186 null pointer, 'pNULL', is a 32 bit pointer; with OFFFFH offset and the 82586 handler's data segment, SEGMT_, as the base.

Put_Cb( ) pushes a free command block back on the list. It does this by placing the cb_tos variable in the returned CB's link pointer field, then updates cb_tos with the pointer to the returned CB.

3.4.2 ISSUING CU COMMANDS–ISSUE_CU_CMD()

This function queues up a command for the 82586 to execute. Since static lists are used, each command has its EL bit set. There is a begin_cbl pointer and an end_cbl pointer to delineate the 82586's static list. If there are no CBs on the list, then begin_cbl is set to pNULL. (Figure 10 illustrates the static list.) Each time a command is issued, a deadman timer is set. When the 82586 interrupts the CPU with a command completed, the deadman timer is reset.

Issue_Cu_Cmd( ) begins by disabling the 82586's interrupt. It then determines whether the list is empty or not. If the list is empty, begin and end pointers are loaded with the CB's address. The CU must then be started. Before a CU_START can be issued, the SCB's cbl_offset field must be loaded with the address of the command, the Wait_Scb( ) function must be called to insure that the SCB is ready to accept a command, and the deadman timer must be initialized. If the list is not empty, then the command block is queued at the end of the list, and the interrupt service routine Isr_586( ), will continue generating CAs for each command linked on the CB list until the list is empty.

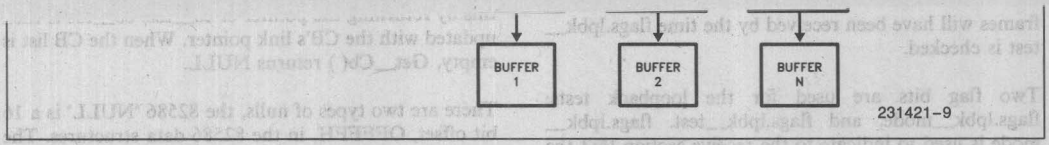


Figure 10. The Static Command Block List

3.4.3 INTERRUPT SERVICE ROUTINE–ISR_586()

Isr_586( ) starts off by saving the interrupts that were generated by the 82586 and acknowledging them. Acknowledgment must be done immediately because if a second interrupt were generated before the acknowledgment, the second interrupt would be missed. The interrupt status is then checked for a receive interrupt and if one occurred the Recv_Int_Processing( ) function is called. After receive processing is check the CPU checks whether a command interrupt occurred. If one did, then the deadman timer is reset and the results of the command are checked. There are only two particular commands which the interrupt results are checked for: Transmit and Diagnose. The Diagnose command needs to be tested to see if it passed, plus the diagnose status flag needs to be set so that the initialization process can continue.

The transmit command status provides network management and station diagnostic information which is useful for the "Network Management" function of the ISO model. The following statistics are gathered in the interrupt routine: good_transmit_cnt, sqe_err_cnt, defer_cnt, no_crs_cnt, underrun_cnt, max_col_cnt. To speed up transmit interrupt processing a flag is tested to determine whether these statistics are desired, if not this section of code is skipped.

The sqe error requires special considerations when used for statistic gathering or diagnostics. The sqe status bit indicates whether the transceiver passed its self test or not. The transceiver executes a self test after each transmission. If the transceiver's self test passed, it will activate the collision signal during the IFS time.

The sqe status bit will be set if the transceiver's self test passed. However if the sqe status bit is not set, the transceiver may still have passed its self test. Several events can prevent the sqe bit from being set. For example, the first transmit command status after power up will not have the sqe bit set because the sqe is always from the previous command. Also if any collisions occur, the sqe bit might not be set. This has to do with the timing of when the sqe signal comes from the transceiver. It is possible that a JAM signal from a remote station can overlap the sqe signal in which case the 82586 will not set the sqe status bit. Therefore the sqe error count should only be recorded when no collisions occur.

One other situation can occur which will prevent the SQE status bit from being set. If transmit command reaches the maximum retry count, the next transmit command's SQE bit will not be set.

The final phase of interrupt command processing determines if another command is linked, and returns the CB to the free command block list. Another command being linked is indicated by the CB link field not being NULL. In this case the deadman timer and the 82586's CU are re-started. If the CB link is NULL, there are no further commands to execute, and begin_cbl is set to pNULL.

3.4.4 SENDING FRAMES–SEND_FRAME (PTBD, PADD)

Send_Frame( ) receives two parameters, a pointer to the first Transmit Buffer Descriptor, and a pointer to the destination address. There may be one or more TBDs attached. The last TBD is indicated by its link

field being NULL and the EOF bit set. It is the responsibility of the ULCS to make sure this is done before calling Send__Frame( ).

Send__Frame( ) begins by trying to obtain a command block. If the free command block list is empty, the send frame function returns with a false result. It is up to the ULCS to either continue attempting transmission or attempt at a later time. The send frame function calculates the length field by summing up the TBDs actual count field. After the length field is determined, send frame checks to see if padding is required. If padding is necessary, Send Frame will change the act count field in the TBD to meet the minimum frame requirements. This technique transmits what ever was in the buffer as padding data. If security is an issue, the padding data in the buffer should be changed.

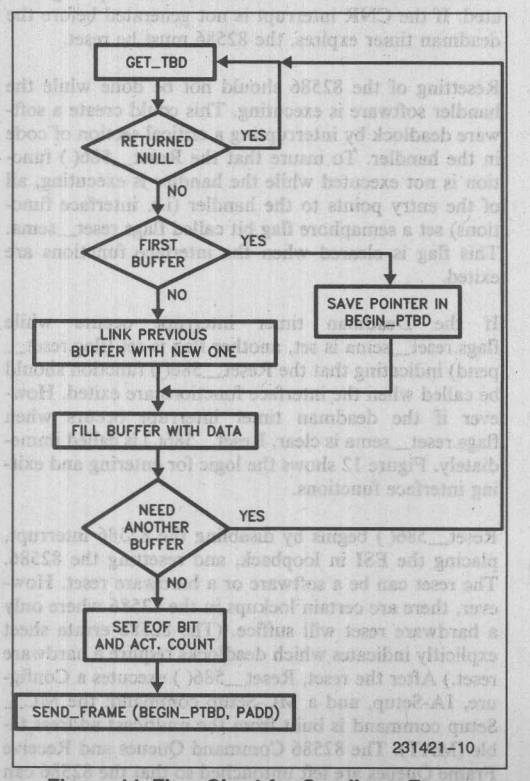


Figure 11. Flow Chart for Sending a Frame

3.4.5 ACCESSING TRANSMIT BUFFERS–GET__TBD() AND PUT__TBD()

Get__Tbd( ) returns a pointer to a free Transmit Buffer Descriptor, and Put__Tbd( ) returns one or more linked Transmit Buffer Descriptors to the free list. The TBD which Get__Tbd( ) allocates has its link pointer set to NULL, and its EOF bit cleared. If another buffer is needed, the link field in the old TBD must be set to point to the new TBD. The last TBD used should have its link pointer set to NULL and its EOF bit set. Figure 11 shows the flow chart of getting buffers and sending a frame.

Put__Tbd (ptbd) is called by the Isr__586( ) function when the 82586 is done transmitting the buffers. A pointer to the first TBD is passed to Put__Tbd( ). Put__Tbd( ) finds the end of the list of TBDs and returns them to the free buffer list.

3.4.6 MULTICAST ADDRESSES

The 82586 handler maintains a table of multicast addresses. Initially this table is empty. To enable a multicast address the Add__Multicast__Address(pma) function is called; to disable a multicast address, Delete__Multicast__Address(pma) function is called. Both functions accept a parameter which points to the multicast address. Add and Delete functions perform linear searches through the Multicast Address Table (MAT).

Add scans the entire MAT once to check if the address being added is a duplicate of one already loaded. Add will not enter a duplicate muiltcast address. If there are no duplicates Add goes to the beginning of the MAT and looks for a free location. If it finds one, it loads the new address into the free location and sets the location status to INUSE. If no free locations are available, Add returns a false result.

Delete looks for a used location in the MAT. When it finds one, it compares the address in the table with the address passed to it. If they match, the location status is set to FREE and a TRUE result is returned. If no match occurs, the result returned is FALSE.

If Add or Delete change the MAT, they update the 82586 by calling Set__Multicast__Address( ). This function executes an 82586 MC Setup command. Set__Mulitcast__Address( ) uses the addresses in the MAT to build the MC Setup command. The MC Setup command is too big to be built from the free CBs. Free CB

command blocks are 18 bytes long, while the MC Setup command can be up to 16,392 bytes. Therefore a separate Multicast Address Command Block (ma_cb) must be allocated and used. The size of the ma_cb and MAT are determined at compile time based on the MULTI_ADDR_CNT constant. The design example allows up to 16 multicast addresses.

Since there is only one ma_cb, and it is not compatible with the other CBs, it must be treated differently. Only one ma_cb can be on the 82586 command list. The ma_cb command word is used as a semaphore. If it is zero, the command is available. If not, Set_Multicast_Address( ) must wait until the ma_cb is free. Also the interrupt routine can't return the ma_cb to the free CB list. It just clears the cmd field, to indicate that ma_cb is available.

The 82586's receiver does not have to be disabled to execute the MC Setup command. If the 82586 is receiving while this command is accessed, the 82586 will finish reception before executing the MC Setup comand. If the MC Setup command is executing, the 82586 automatically ignores incoming frames until the MC Setup is completed. Therefore multicast addresses can be added and deleted on the fly.

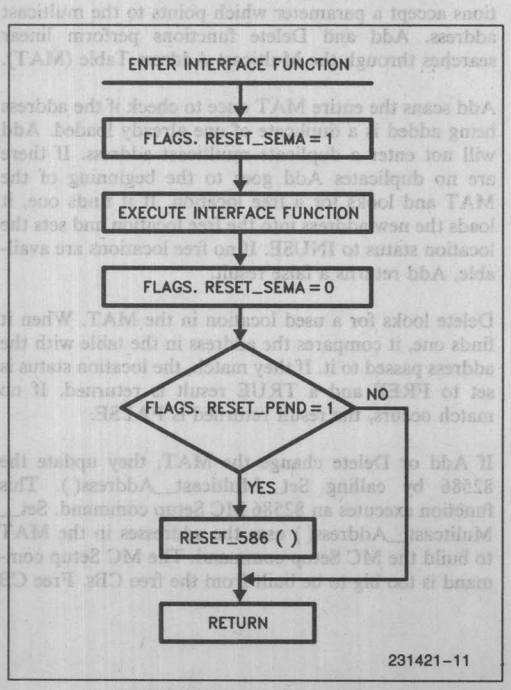


Figure 12. Reset Semaphore

3.4.7 RESETTING THE 82586–RESET_586()

The 82586 rarely if ever locks up in a well behaved network; (i.e. one that obeys IEEE 802.3 specifications). The lock-ups identified were artificially created and would normally not occur. This data link driver has been tested in an 8 station network under various loading conditions. No lock-ups occurred under any of the data link drivers test conditions. However the reset software has been tested by simulating a lockup. This can be done by having the 82586 transmit, and disabling the CTS pin for a time longer than the deadman timer.

An 82586 deadlock is not a fatal error. The handler is designed to recover from this problem. As mentioned before, each time the 82586 is given a CA to begin executing a command, a deadman timer is set. The deadman timer is reset when a CNR interrupt is generated. If the CNR interrupt is not generated before the deadman timer expires, the 82586 must be reset.

Resetting of the 82586 should not be done while the handler software is executing. This could create a software deadlock by interrupting a critical section of code in the handler. To insure that the Reset_586( ) function is not executed while the handler is executing, all of the entry points to the handler (i.e. interface functions) set a semaphore flag bit called flags.reset_sema. This flag is cleared when the interface functions are exited.

If the Deadman timer interrupt occurs while flags.reset_sema is set, another flag is set (flag.reset_pend) indicating that the Reset_586( ) function should be called when the interface functions are exited. However if the deadman timer interrupt occurs when flags.reset_sema is clear, Reset_586( ) is called immediately. Figure 12 shows the logic for entering and exiting interface functions.

Reset_586( ) begins by disabling the 82586 interrupt, placing the ESI in loopback, and resetting the 82586. The reset can be a software or a hardware reset. However, there are certain lockups in the 82586 where only a hardware reset will suffice. (The 82586 errata sheet explicitly indicates which deadlocks require a hardware reset.) After the reset, Reset_586( ) executes a Configure, IA-Setup, and a MC-Setup command; the MC_Setup command is built from the multicast address table (MAT). The 82586 Command Queues and Receive Frame Queues are left untouched so that the 82586 can continue executing where it left off before the deadlock. This way no frames or commands are lost. This requires that a separate reset CB and reset Multicast CB is used, because other CBs already in use cannot be disturbed.

3.5 Receive Frame Processing

The following functions are used for Receive Frame Processing:

Recv_Int_Processing( )	Called by Isr_586( ) to remove FDs and RBDs from the 82586's RFA
Recv_Frame (pfd)	Called by Recv_Int_Processing( ). This function resides in the ULCS
Check_Multicast (pfd)	Used for perfect Multicast filtering
Put_Free_Rfa (pfd)	Returns FDs and RBDs to the 82586's RFA
Ru_Start( )	Restarts the RU when in the IDLE or No Resources state.

3.5.1 RECEIVE INTERRUPT PROCESSING–RECV_INT_PROCESSING()

The Recv_Int_Processing( ) function is called by Isr_586( ) when the FR bit in the SCB is set. The Recv_Int_Processing( ) function checks whether any FDs and RBDs on the free list have been used by the 82586. If they have, Recv_Int_Processing( ) removes the used FDs and RBDs from the free list, and passes them to the ULCS.

The Recv_Int_Processing( ) function is a loop where each pass removes a frame from the 82586's RFA. When there are no more used FDs and RBDs on the RFA, the function calls RU_Start( ), then returns to Isr_586( ). The first part of the loop checks to see if the C bit in the first FD of the free FD list is set. If the C bit is set, the function determines if one or more RBDs are attached. If there are RBDs attached, the end of the RBD list is found. The last RBD's link field is used to update begin_rbd pointer, and then it's set to NULL.

After the receive frame has been delineated from the RFA, some information about the frame is needed to determine which function to pass it to. Since the save bad frame configure bit is not set, the only bad frame on the list could be an out of resource frame. An out of resource frame is returned to the RFA by calling Put_Free_RFA (pfd). If the flags.lpbk_mode bit is set, the frame is given to the loopback check function. If the destination address of the frame indicates a multicast, the check multicast function is called. If the frame has passed all of the above tests and still has not been returned, it is passed to the Recv_Frame( ) function which resides in the ULCS.

Check_Multicast (pfd) determines whether the multicast address received is in the multicast address table. This is necessary because the 82586 does not have perfect multicast address filtering. Check_Multicast does a byte by byte comparison of the destination address with the addresses in the multicast address table. If no match occurs, it returns false, and Recv_Int_Processing calls Put_Free_RFA( ) to return the frame to the RFA. If there is a match, Check_Multicast( ) returns TRUE and Recv_Int_Processing( ) calls Recv_Frame( ), passing the pointer to the FD of the frame received.

3.5.2 RETURNING FDs AND RBDs–PUT_FREE_RFA (pfd)

Put_Free_RFA combines Supply_FD and Supply_RBD algorithms described in "Programming the 82586" into one function. The begin and end pointers delineate what the CPU believes is the beginning and end of the free list. The decision of whether to restart the RU is made when examining both the free FD list and the free RBD list. This is why two ru_start_flags are used, one for the FD list and one for the RBD list. Both flags are initialized to FALSE.

The function starts off by initializing the FD so that the EL bit is set, the status is 0, and the FD link field is NULL. The rbd pointer is saved before the rbd pointer field in the FD is set to NULL. The free FD list is examined and if it's empty, begin—fd and end—fd are loaded with the address of the FD being returned. In this case the RU should not be restarted, because there is only one FD on the free list. If the free FD list is not empty, the FD being returned is placed on the end of the list, the end pointer is updated, and the RU start flag is set TRUE.

To begin the RBD list processing the end of the returned RBD list is determined, and this last RBD's EL bit is set. If the free RBD list is empty, the returned RBD list becomes the free RBD list. If there is more than one RBD on the returned list, the ru start flag is set TRUE. If the free RBD list is not empty, the returned RBD list is appended on the end of the free list, the end—rbd pointer is updated, and the ru start flag is set TRUE.

The last part of Put_Free_RFA( ) is to determine whether to call RU_Start( ). Both ru start flags are ANDed together, and if the result is TRUE, the Ru_Start( ) function is called.

3.5.3 RESTARTING THE RECEIVE UNIT–RU_START()

The Ru_Start( ) function checks two things before it decides to restart the RU. The first thing it checks is whether the RU is already READY. If it is, there is no reason to restart it. If the RU is IDLE or in NO_RESOURCES, then the second thing to check is whether the first free FD on the free FD list has its C bit set. If it does, then the RU should not be restarted. The reason is that the free FD list should only contain free FDs

4.0 LOGICAL LINK CONTROL

The IEEE 802.2 LLC function completes the Data Link Layer of the OSI model. The LLC module in this design example implements a class 1 level of service which provides a connectionless datagram interface. Several data link users or processes can run on top of the data link layer. Each user is identified by a link service access point (LSAP). Communication between data link users is via LSAPs. An LSAP is an address that identifies a specific user process or another layer

Figure 14 illustrates the relationship between the Station Component and the SAP components. (The SAP components are user processes.) The Station Component receives all of the good frames from the Handler and checks the DSAP address. If the DSAP address is 0, then the frame is addressed to the Station Component and a Station Component Response is generated. If the DSAP address is on the active DSAP list, then the Station Component passes the frame to the addressed SAP. If the DSAP address is unknown, the frame is returned to the handler.

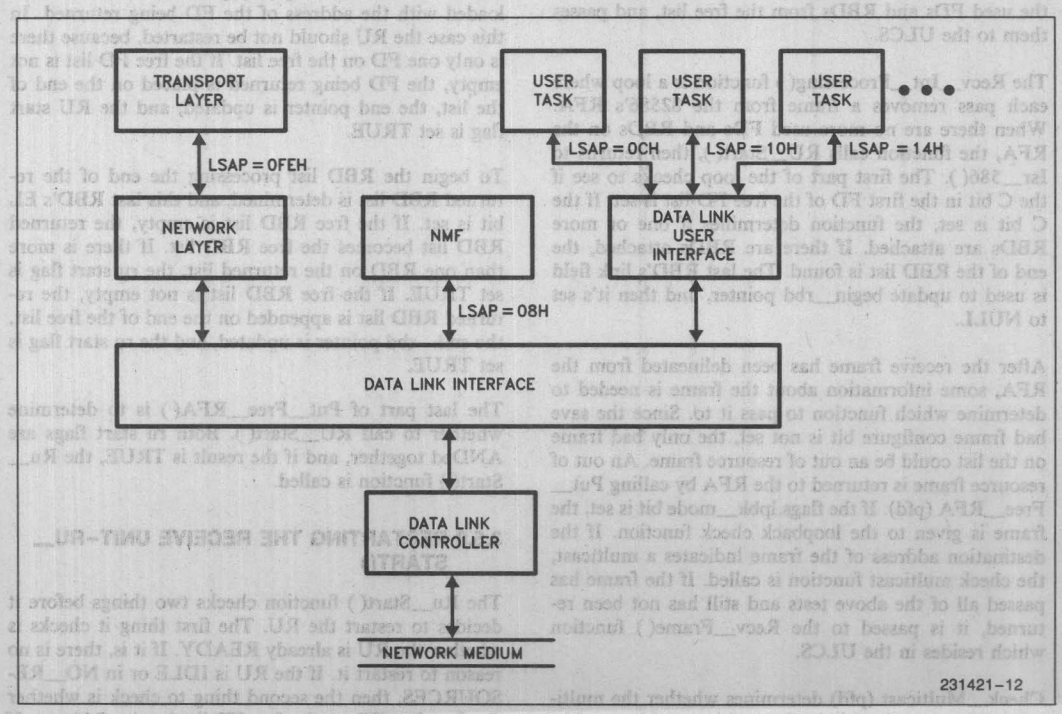


Figure 13. Data Link Interface

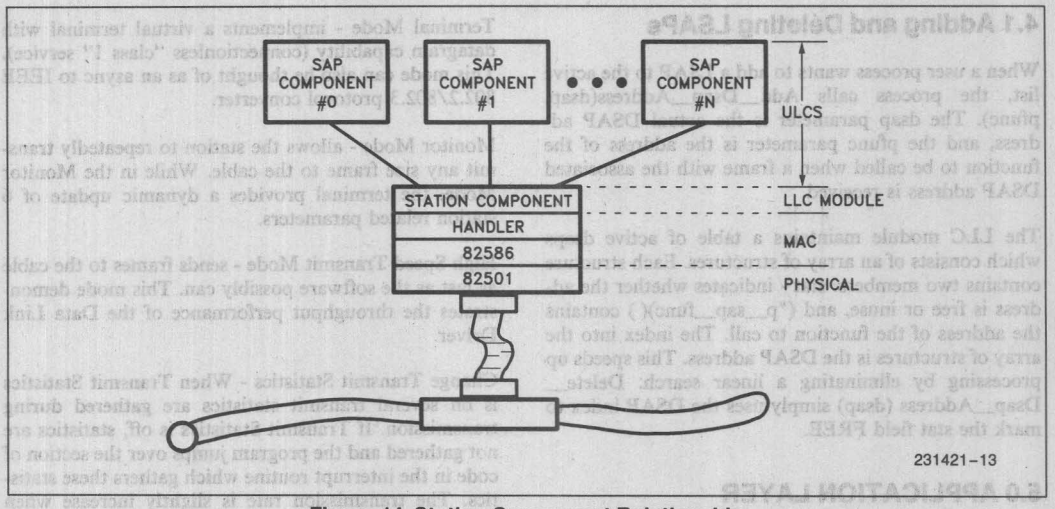


Figure 14. Station Component Relationship

There are 3 commands and 2 responses which the class 1 LLC layer must implement. Figure 15 shows IEEE 802.2 Class 1 commands and responses and Figure 16 shows the IEEE 802.2 Class 1 frame format.

Commands	Responses	Description
UI		Unnumbered Information
XID	XID	Exchange ID
TEST	TEST	Remote Loopback

Figure 15. IEEE 802.2 Class 1, Type 1 Commands and Responses

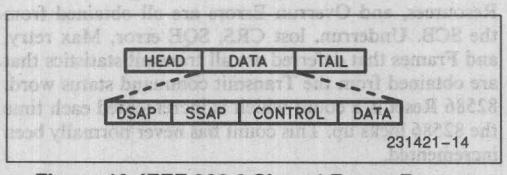


Figure 16. IEEE 802.2 Class 1 Frame Format

From Figure 15 it can be seen that there are no LLC class 1 UI responses because information frames are not acknowledged at the data link level. The only command frames that may require responses are XID and TEST. If a command frame is addressed to the Station Component, it checks the control field to see what type of frame it is. If it's an XID frame, the Station Component responds with a class 1 XID response frame. If it's a TEST frame, the Station Component responds with a TEST frame, echoing back the data it received. In both cases, the response frame is addressed to the source of the command frame.

Any frames addressed to active SAPs are passed directly to them. The Station Component will not respond to SAP addressed frames. Therefore it is the responsibility of the SAPs to recognize and respond to frames addressed to them. When a SAP transmits a frame, it builds the IEEE 802.2 frame itself and calls the Handler's Send_Frame( ) function directly. The LLC module is not used for SAP frame transmission. The only functions which the LLC module implement are the dynamic addition and deletion of DSAPs, multiplexing the frames to user SAPs, and the Station Component command recognition and responses. This is one implementation of the IEEE 802.2 standard. Other implementations may have the LLC module do more functions, such as SAP command recognitions and responses. A list of the functions included in the LLC module is as follows:

LLC Functions	Description
Init_Llc( )	Initializes the DSAP address table and calls Init_586( )
Add_Dsap_ Address (dsap, pfunc)	Add a DSAP address to the active list dsap - DSAP address pfunc - pointer to the SAP function
Delete—Dsap— Address (dsap)	Delete a DSAP address dsap - DSAP address
Recv—Frame (pfd)	Receives a frame from the 82586 Handler pfd - Frame Descriptor Pointer
Station—Component— Response (pfd)	Generates a response to a frame addressed to the Station Component pfd - Frame Descriptor Pointer

4.1 Adding and Deleting LSAPs

When a user process wants to add a LSAP to the active list, the process calls Add_Dsap_Address(dsap, pfunc). The dsap parameter is the actual DSAP address, and the pfunc parameter is the address of the function to be called when a frame with the associated DSAP address is received.

The LLC module maintains a table of active dsaps which consists of an array of structures. Each structure contains two members: stat - indicates whether the address is free or inuse, and (*p_sap_func)( ) contains the address of the function to call. The index into the array of structures is the DSAP address. This speeds up processing by eliminating a linear search. Delete_Dsap_Address (dsap) simply uses the DSAP index to mark the stat field FREE.

5.0 APPLICATION LAYER

For most networks the application layer resides on top of several other layers referred to here as ULCS. These other layers in the OSI model run from the network layer through the presentation layer. The implementation of the ULCS layers is beyond the scope of this application note, however Intel provides these layers as well as the data link layer with the OpenNET product line. For the purpose of this application note the application layer resides on top of the data link layer and its use is to demonstrate, exercise and test the data link layer design example.

There can be several processes sitting on top of the data link layer. Each process appears as a SAP to the data link. The UAP module, which implements the application layer, is the only SAP residing on top of the data link layer in this application example. Other SAPs could certainly be added such as additional "connectionless" terminals, a networking gateway, or a transport layer, however in the interest of time this was not done.

5.1 Application Layer Human Interface

The UAP provides a menu driven human interface via an async terminal connected to port B on the iSBC 186/51 board. The menu of the commands is listed in Figure 17 along with a description that follows:

```
T - Terminal Mode                    M - Monitor Mode
X - High Speed Transmit Mode         V - Change Transmit Statistics
P - Print All Counters               C - Clear All Counters
A - Add a Multicast Address          Z - Delete a Multicast Address
S - Change the SSAP Address          D - Change the DSAP Address
N - Change Destination Node Address  L - Print All Addresses
R - Re-Initialize the Data Link      B - Change the Number Base
```

Figure 17. Menu of Data Link Driver Commands

Terminal Mode - implements a virtual terminal with datagram capability (connectionless "class 1" service). This mode can also be thought of as an async to IEEE 802.2/802.3 protocol converter.

Monitor Mode - allows the station to repeatedly transmit any size frame to the cable. While in the Monitor Mode, the terminal provides a dynamic update of 6 station related parameters.

High Speed Transmit Mode - sends frames to the cable as fast as the software possibly can. This mode demonstrates the throughput performance of the Data Link Driver.

Change Transmit Statistics - When Transmit Statistics is on several transmit statistics are gathered during transmission. If Transmit Statistics is off, statistics are not gathered and the program jumps over the section of code in the interrupt routine which gathers these statistics. The transmission rate is slightly increase when Transmit Statistics is off.

Print All Counters - Provides current information on the following counters.

Good frames transmitted:
Good frames received:
CRC errors received:
Alignment errors received:
Out of Resource frames:
Receiver overrun frames:

Each time a frame has been successfully transmitted the Good frames transmitted count is incremented. The same holds true for reception. CRC, Alignment, Out of Resources, and Overrun Errors are all obtained from the SCB. Underrun, lost CRS, SQE error, Max retry, and Frames that deferred are all transmit statistics that are obtained from the Transmit command status word. 82586 Reset is a count which is incremented each time the 82586 locks up. This count has never normally been incremented.

Clear All Counters - Resets all of the counters.

Add/Delete Multicast Address - Adds and Deletes Multicast Addresses.

Change SSAP Address - Deletes the previous SSAP and adds a new one to the active list. The SSAP in this case is this stations LSAP. When a frame is received, the DSAP address in the frame received is compared with any active LSAPs on the list. The SSAP is also used in the SSAP field of all transmitted frames.

Change DSAP Address - Delete the old DSAP and add a new one. The DSAP is the address of the LSAP which all transmit frames are sent to.

Change Destination Node Address - Address a new node.

Print All Addresses - Display on the terminal the station address, destination address, SSAP, DSAP, and all multicast addresses.

Re-initialize Data Link - This causes the Data Link to completely reinitialize itself. The 82586 is reset and reinitialized, and the selftest diagnostic and loopback tests are executed. The results of the diagnostics are printed on the terminal. The possible output messages from the 82586 selftest diagnostics are:

Passed Diagnostic Self Tests

Failed: Self Test Diagnose Command

Failed: Internal Loopback Self Test

Failed: External Loopback Self Test

Failed: External Loopback Through Transceiver Self Test

Change Base - Allows all numbers to be displayed in Hex or Decimal.

5.2 A Sample Session

The following text was taken directly from running the Data Link software on a 186/51 board. It begins with the iSDM monitor signing on and continues into executing the Data Link Driver software.

```
iSDM 86 Monitor, V1.0

Copyright 1983 Intel Corporation

.G D000:6

                ****************************************************************
                *                                                              *
                * 82586 IEEE 802.2/802.3 Compatible Data Link Driver           *
                *                                                              *
                ****************************************************************

Passed Diagnostic Self Tests

Enter the Address of the Destination Node in Hex -> 00AA0000179E

Enter this Station's LSAP in Hex -> 20

Enter the Destination Node's LSAP in Hex -> 20

Do you want to Load any Multicast Addresses? (Y or N) -> Y

Enter the Multicast Address in Hex -> 00AA00111111

Would you like to add another Multicast Address? (Y or N) -> N

This Station's Host Address is: 00AA00001868

The Address of the Destination Node is: 00AA0000179E

This Station's LSAP Address is: 20

The Address of the Destination LSAP is: 20

The following Multicast Addresses are enabled: 00AA00111111
```

```
Enter a command, type H for Help -> P

Good frames transmitted:     24      Good frames received:     1
CRC errors received:          0      Alignment errors received: 0
Out of Resource frames:       0      Receiver overrun frames:   0
82586 Reset:                  0      Transmit underrun frames:  0
Lost CRS:                     0      SQE errors:                9
Maximum retry:                0      Frames that deferred:      4

Enter a command, type H for Help --> T

Would you like the local echo on? (Y or N) --> Y

This program will now enter the terminal mode.

Press ^C then CR to return back to the menu

Hello this is a test.

/*^C CR */

Enter a command, type H for Help --> M

Do you want this station to transmit? (Y or N) --> Y

Enter the number of data bytes in the frame --> 1500

Hit any key to exit Monitor Mode.

 # of Good      # of Good        CRC        Alignment       No        Receive
   Frames        Frames         Errors       Errors      Resource    Overrun
Transmitted     Received                                  Errors      Errors
      32            0          00000        00000        00000       00000

/* CR */

Enter a command, type H for Help --> X

Hit any key to exit High Speed Transmit Mode.

/* CR */

Enter a command, type H for Help --> R
Passed Diagnostic Self Tests
```

5.3 Terminal Mode

The Terminal mode buffers characters received from the terminal and sends them in a frame to the cable. When a frame is received from the cable, data is extracted and sent to the terminal. One of three events initiate the UAP to send a frame providing there is data to send: buffering more than 1500 bytes, receiving a Carriage Return from the terminal, or receiving an interrupt from the virtual terminal timer.

The virtual terminal timer employs timer 1 in the 80130 to cause an interrupt every .125 seconds. Each time the interrupt occurs the software checks to see if it received one or more characters from the terminal. If it did, then it sends the characters in a frame.

The interface to the async terminal is a 256 byte software FIFO. Since the terminal communication is full duplex, there are two half duplex FIFOs: a Transmit FIFO and a Receive FIFO. Each FIFO uses two functions for I/O: Fifo_In( ) and Fifo_Out( ). A block diagram is displayed in Figure 18.

The serial I/O for the async terminal interface is always polled except in the Terminal mode where it is interrupt driven. The Terminal mode begins by enabling the 8274 receive interrupt but leaves the 8274 transmit interrupt disabled. This way any characters received from the terminal will cause an interrupt. These characters are then placed in the Transmit FIFO. The only time the 8274 transmit interrupt is enabled is when the Receive FIFO has data in it. The receive FIFO is filled from frames being received from the cable. Each time a transmit interrupt occurs a byte is removed from the Receive FIFO and written to the 8274. When the Receive FIFO empties, the 8274 transmit interrupt is disabled.

The flow control implemented for the terminal interface is via RTS and CTS. When the Transmit FIFO is full, RTS goes inactive preventing further reception of characters (see Table 1). If the Receive FIFO is full, receive frames are lost because there is no way for the data link using class 1 service to communicate to the remote station that the buffers are full. Lost receive frames are accounted for by the Out of Resources Frame counter.

The Async Terminal bit rate sets the throughput capability of the station in the terminal mode because the bottle neck for this network is the RS232 interface. Using this fact a simple test was conducted to verify the data link driver's capability of switching between the receiver's No Resource state and the Ready State. For example if station B is sending frames in the High Speed Transmit mode to station A which is in the Terminal mode, frames will be lost in station A. Under these circumstances station A's receiver will be switching from Ready state to Out of Resources state. The sum of Good frames received plus Out of Resource frames from station A should equal Good frames transmitted from station B; unless there were any underruns or overruns.

Table 1. FIFO State Table

Function	Present State	Next State	Action
FIFO_T_IN( )	EMPTY	IN USE	Start Filling Transmit Buffer
	IN USE	FULL	Shut Off RTS
FIFO_T_OUT( )	FULL	IN USE	Enable RTS
	IN USE	EMPTY	Stop Filling Transmit Buffer
FIFO_R_IN( )	EMPTY	IN USE	Turn on TxInt
	IN USE	FULL	Stop Filling FIFO from Receive Buffer
FIFO_R_OUT( )	FULL	IN USE	Start Filling FIFO from Receive Buffer
	IN USE	EMPTY	Turn Off TxInt

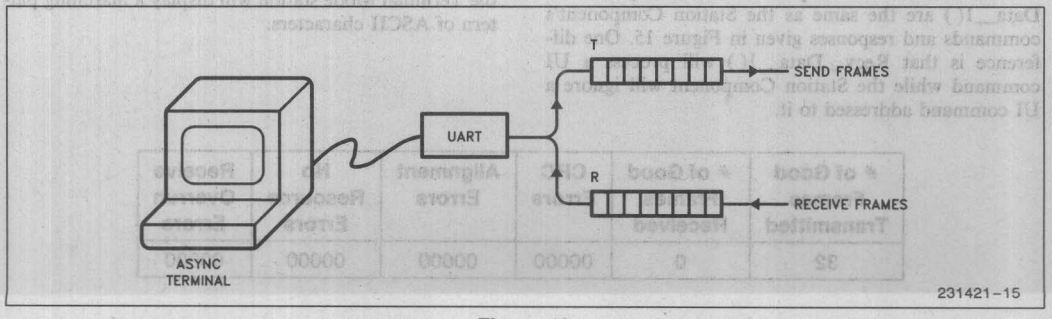


Figure 18

5.3.1 SENDING FRAMES

The Terminal Mode is entered when the Terminal_Mode( ) function is called from the Menu interface. The Terminal_Mode( ) function is one big loop, where each pass sends a frame. Receiving frames in the Terminal Mode is handled on an interrupt driven basis which will be discussed next.

The loop begins by getting a TBD from the 82586 handler. The first three bytes of the first buffer are loaded with the IEEE 802.2 header information. The loop then waits for the Transmit FIFO to become not EMPTY, at which point a byte is removed from the Transmit FIFO and placed in the TBD. After each byte is removed from the Transmit FIFO several conditions are tested to determine whether the frame needs to be transmitted, or whether a new buffer must be obtained. A frame needs to be transmitted if: a Carriage Return is received, the maximum frame length is reached, or the send_frame flag is set by the virtual terminal timer. A new buffer must be obtained if none of the above is true and the max buffer size is reached.

If a frame needs to be sent the last TBD's EOP bit is set and its buffer count is updated. The 82586 Handler's Send_Frame( ) function is called to transmit the frame, and continues to be called until the function returns TRUE.

The loop is repeated until a ^C followed by a Carriage Return is recieved.

5.3.2 RECEIVING FRAMES

Upon initialization the UAP module calls the Add_Dsap_Address(dsap, pfunc) function in the LLC module. This function adds the UAP's LSAP to the active list. The pfunc parameter is the address of the function to call when a frame has been received with the UAP's LSAP address. This function is Recv_Data_1( ). Recv_Data_1( ) looks at the control field of the frame received and determines the action required.

The commands and responses handled by Recv_Data_1( ) are the same as the Station Component's commands and responses given in Figure 15. One difference is that Recv_Data_1( ) will process a UI command while the Station Component will ignore a UI command addressed to it.

Recv_Data_1( ) will discard any UI frames received unless it is in the Terminal Mode. When in the Terminal Mode, Recv_Data_1( ) skips over the IEEE 802.2 header information and uses the length field to determine the number of bytes to place in the Receive FIFO. Before a byte is placed in the FIFO, the FIFO status is checked to make sure it is not full. Recv_Data_1( ) will move all of the data from the frame into the Receive FIFO before returning.

When a frame is received by the 82586 handler an interrupt is generated. While in the 82586 interrupt routine the receive frame is passed to the LLC layer and then to the UAP layer where the data is placed in the Receive FIFO by Recv_Octal_Data_1( ). Since Recv_Data_1( ) will not return until all of the data from the frame has been moved into the Receive FIFO, the 8274 transmit interrupt must be nested at a higher priority than the 82586 interrupt to prevent a software lock. For example if a frame is received which has more than 256 bytes of data, the Receive FIFO will fill up. The only way it can empty is if the 8274 interrupt can nest the 82586 interrupt service routine. If the 8274 could not interrupt the 82586 ISR then the software would be stuck in Recv_Data_1( ) waiting for the FIFO to empty.

5.4 Monitor Mode

The Monitor Mode dynamically updates 6 station related parameters on the terminal as shown below.

The Monitor_Mode( ) function consists of one loop. During each pass through the loop the counters are updated, and a frame is sent. Any size frame can be transmitted up to a size of the maximum number of transmit buffers available. Frame sizes less than the minimum frame length are automatically padded by the 82586 Handler.

The data in the frames transmitted in the Monitor Mode are loaded with all the printable ASCII characters. This way when one station is in the Monitor Mode transmitting to another station in the Terminal Mode, the Terminal Mode station will display a marching pattern of ASCII characters.

# of Good Frames Transmitted	# of Good Frames Received	CRC Errors	Alignment Errors	No Resource Errors	Receive Overrun Errors
32	0	00000	00000	00000	00000

5.5 High Speed Transmit Mode

The High Speed Transmit Mode demonstrates the throughput performance of the 82586 Handler. The Hs_Xmit_Mode( ) function operates in a tight loop which gets a TBD, sets the EOF bit, and calls Send_Frame( ). The flow chart for this loop is shown in Figure 19.

The loop is exited when a character is received from the terminal. Rather than polling the 8274 for a receive buffer full status, the 8274's receive interrupt is used. When the Hs_Xmit_Mode( ) function is entered, the hs_stat flag is set true. If the 8274 receive interrupt occurs, the hs_stat flag is set false. This way the loop only has to test the hs_stat flag rather than calling inb( ) function each pass through the loop to determine whether a character has been received.

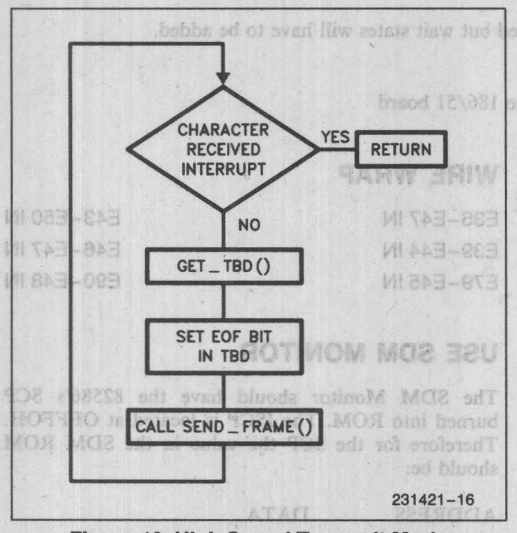


Figure 19. High Speed Transmit Mode Flow Chart

The performance measured on an 8 MHz 186/51 board is 593 frames per second. The bottle neck in the throughput is the software and not the 82586. The size of the buffer is not relevant to the transmit frame rate. Whether the buffer size is 128 bytes or 1500 bytes, linked or not, the frame rate is still the same. Therefore assuming a 1500 byte buffer at 593 frames per second, the effective data rate is 889,500 bytes per second.

This can easily be demonstrated by using two 186/51 boards running the Data Link software. The receiving stations counters should be cleared then placed in the Monitor mode. When placing it in the monitor mode, transmission should not be enabled. When the other station is placed in the High Speed Transmit Mode a timer should be started. One can use a stop watch to determine the time interval for transmission. The frame rate is determined by dividing the number of frames received in the Monitor station by the time interval of transmission.

JUMPERS REQUIRED

Jumper the 186/51 board for 16K byte PROMs in U34 and U39 Table 2–5 in 186/51 HARDWARE REFERENCE MANUAL (Rev-001)

186/51(ES)	186/51 (S)/186/51
E151–E152 OUT	E199–E203 OUT
E152–E150 IN	E203–E191 IN
E94–E95 IN	E120–E119 IN
E100–E106 IN	E116–E112 IN
E107–E113 IN	E111–E107 IN
E133-E134 IN	E94–E93 IN

also change interrupt priority jumpers - switch 8274 and 82586 interrupt priorities

E36–E44 OUT	E43–E47 OUT
E39–E47 OUT	E46–ES0 OUT
E37–E45 OUT	E44–E48 OUT

WIRE WRAP

E36–E47 IN	E43–E50 IN
E39–E44 IN	E46–E47 IN
E79–E45 IN	E90–E48 IN

USE SDM MONITOR

The SDM Monitor should have the 82586's SCP burned into ROM. The ISCP is located at OFFFOH. Therefore for the SCP the value in the SDM ROM should be:

ADDRESS	DATA
FFFF6H	XXOOH
FFFF8H	XXXXH
FFFFAH	XXXXH
FFFFCH	FFFOH
FFFFEH	XXOOH

To run the program begin execution at 0D000:6H

```
I.E. G D000:6

GOOD LUCK!

**********    submit file for compiling one module:    **********

run

cc86.86  :F6:%0 LARGE ROM DEBUG DEFINE(DEBUG) include(:F6:)

exit

**********    submit file for linking and locating:    **********

run

link86      :F6:assy.obj, :F6:dld.obj, :F6:llc.obj, &

:F6:uap.obj, lclib.lib to :F6:dld.lnk segsize(stack(4000h)) notype

loc86 :F6:dld.lnk to :F6:dld.loc&

initcode (0D0000H) start(begin) order(classes(data, stack, code)) &
addresses(classes(data(3000H), stack(0CB00H), code(0D0020H)))

oh86 :F6:dld.loc to :F6:dld.rom

exit

**********    submit file for burning EPROMs using IPPS:    **********

ipps

i 86

f :F6:dld.rom (0d0000h)

3

2

1

0 to :F6:lo.byt

y

1 to :F6:hi.byt

y


t 27128

9

c :F6:lo.byt t p

n

C :F6:hi.byt t p

n

exit
```

```
/PCD/USR/CHUCK/CSRC/DLD.H

/*************************************************************************
*                                                                        *
*                        82586 Structures and Constants                  *
*                                                                        *
*************************************************************************/

/* general purpose constants */

#define INUSE    0
#define EMPTY    1
#define FULL     2
#define FREE     1
#define TRUE     1
#define FALSE    0
#define NULL     0xFFFF

/* Define Data Structures */

#define RBUF_SIZE        128 /* receive buffer size */
#define TBUF_SIZE        128 /* transmit buffer size */
#define ADD_LEN          6
#define MULTI_ADDR_CNT   16

typedef unsigned short int u_short;

/* results from Test_Link(): loaded into Self_Test char */

#define PASSED                  0
#define FAILED_DIAGNOSE         1
#define FAILED_LPBK_INTERNAL    2
#define FAILED_LPBK_EXTERNAL    3
#define FAILED_LPBK_TRANSCEIVER 4

/* Frame Commands */
#define     UI       0x03    /* Unnumbered Information Frame */
#define     XID      0xAF    /* Exchange Identification */
#define     TEST     0xE3    /* Remote Loopback Test */
#define     P_F_BIT  0x10    /* Poll/Final Bit Position */
#define     C_R_BIT  0x01    /* Command/Response bit in SSAP */


#define     DSAP_CNT     8   /* Number of allowable DSAPs; must be a multiple
                                of 2**N, and DSAP addresses assigned must be
                                divisible by 2**(8-N).
                                (i.e. the N LSBs must be 0) */

#define     DSAP_SHIFT   5   /* DSAP_SHIFTS must equal 8-N */


#define     XID_LENGTH   6   /* Number of Info bytes for XID Response frame */


/* System Configuration Pointer SCP */

   struct SCP  {
               u_short sysbus; /* 82586 bus width, 0 - 16 bits
                                  1 - 8 bits */
```

231421-17

```
/PCO/USR/CHUCK/CSRC/DLD.H

                u_short junk[2];
                u_short iscpl;  /* lower 16 bits of iscp address */
                u_short iscph;  /* upper 8 bits of iscp address */
               };

/* Intermediate System Configuration Pointer ISCP */

   struct ISCP {
                u_short busy ;  /*set to 1 by cpu before its first CA,
                                 cleared by 82586 after reading */
                u_short offset  ;   /* offset of system control block */
                u_short base1 ; /* base of system control block */
                u_short base2 ;
               } ;

/* System Control Block SCB */

   struct SCB  {
                u_short stat;          /* Status word */
                u_short cmd;           /* Command word */
                u_short cbl_offset;  /* Offset of first command block in CBL */
                u_short rfa_offset;  /* Offset of first frame descriptor in RFA */
                u_short crc_errs;    /* CRC errors accumulated */
                u_short aln_errs;    /* Alignment errors */
                u_short rsc_errs;    /* Frames lost because of no Resources */
                u_short ovr_errs;    /* Overrun errors */
               };

/*  Command Block */

   struct  CB  {
                u_short stat;    /* Status of Command */
                u_short cmd;     /* Command */
                u_short link;    /* link field */
                u_short parm1;   /* Parameters */
                u_short parm2;
                u_short parm3;
                u_short parm4;
                u_short parm5;
                u_short parm6;
          };


/*  Multicast Address Command Block MA_CB */

   struct MA_CB{
                u_short stat;    /* Status of Command */
                u_short cmd;     /* Command */
                u_short link;    /* Link field */
                u_short mc_cnt; /* Number of MC addresses */
                char mc_addr[ADD_LEN*MULTI_ADDR_CNT]; /* MC address area */
               };


/* Transmit Buffer Descriptor TBD */

   struct TBD  {
```

231421-18

```
int r
                u_short rbd_offset;          /* Receive buffer descriptor offset */
                char dest_addr[ADD_LEN];/*Destination address */
                char src_addr[ADD_LEN]; /* Source address */
                u_short length;              /* Length field */
               };

/* Receive Buffer Descriptor RBD */

   struct RBD {
                u_short act_cnt;          /* Actual number of bytes received */
                u_short link;             /* Offset to next RBD */
                u_short buff_l;           /* Lower 16 bits of buffer address */
                u_short buff_h;           /* upper 8 bits of buffer address */
                u_short size;             /* size of buffer */
                struct RB *buff_ptr;      /* not used by the 586. used by the
                                          software to save address translation
                                          routine. */
               };


/* Receive Buffers */

   struct RB    {
                 char data[RBUF_SIZE];
                };


struct  FRAME_STRUCT
        {
            unsigned char   dsap;    /* Destination Service Access Point */
            unsigned char   ssap;    /* Source Service Access Point */
            unsigned char   cmd;     /* ISO Data Link Command */
        };

    /* LSAP Address Table */
struct  LAT {
                 char    stat;            /* INUSE or FREE */
```

231421-19

```
/PCO/USR/CHUCK/CSRC/DLD.H

                int     (*p_sap_func)();/* Pointer to LSAP function; associated
                                           with dsap address */
        };

struct MAT {                        /* Multicast Address Table */
        char    stat;               /* INUSE or FREE */
        char    addr[ADD_LEN];      /* actual mc address */
        };

/* general purpose flags */

struct FLAGS {
    unsigned diag_done : 1 ;    /* diagnose command complete */
    unsigned stat_on : 1 ;      /* network diagnostic statistics on/off */
    unsigned reset_sema: 1 ;    /* don't reset when this bit is set */
    unsigned reset_pend: 1 ;    /* reset when this bit is set */
    unsigned lpbk_test: 1 ;     /* loopback test flag */
    unsigned lpbk_mode: 1 ;     /* loopback mode on/off */
      } ;

/* General purpose bits */

#define ELBIT   0x8000
#define EOFBIT  0x8000
#define SBIT    0x4000
#define IBIT    0x2000
#define CBIT    0x8000
#define BBIT    0x4000
#define OKBIT   0x2000

/* SCB patterns */

#define CX          0x8000
#define FR          0x4000
#define CNA         0x2000
#define RNR         0x1000
#define RESET       0x0080
#define CU_START    0x0100
#define RU_START    0x0010
#define RU_ABORT    0x0040
#define CU_MASK     0x0700
#define RU_MASK     0x0070
#define RU_READY    0x0040

/* 82586 Commands */

#define NOP         0x0000
#define IA          0x0001
#define CONFIGURE   0x0002
#define MC_SETUP    0x0003
#define TRANSMIT    0x0004
#define TDR         0x0005
#define DUMP        0x0006
#define DIAGNOSE    0x0007
```

231421-20

```
/PCO/USR/CHUCK/CSRC/DLD.H

/* 82586 Command and Status Masks */

#define CMD_MASK          0x0007
#define NOERRBIT          0x2000
#define COLLMASK          0x000F
#define DEFERMASK         0x0080
#define NOCRSMASK         0x0400
#define UNDERRUNMASK      0x0100
#define SQEMASK           0x0040
#define MAXCOLMASK        0x0020
#define OUT_OF_RESOURCES 0x0200

/* Configure Parameters */

#define FIFO_LIM    0x0800      /* use FIFO lim of 8 */
#define BYTE_CNT    0x000B
#define SRDY        0x0040
#define SAV_BF      0x0080
#define ADDR_LEN    0x0600      /* address length of 6 bytes */
#define AC_LOC      0x0800
#define PREAM_LEN   0x2000      /* preamble length of 8 bytes */
#define INT_LPBCK   0x4000
#define EXT_LPBCK   0x8000
#define LIN_PRIO    0x0000      /* no priority */
#define ACR         0x0000
#define BOF_MET     0x0080
#define IFS         0x6000      /* IFS time 9.6 usec */
#define SLOT_TIME   0x0200      /* slot time 51.2 usec */
#define RETRY_NUM   0xF000      /* retry number 15 */
#define PRM         0x0001
#define BC_DIS      0x0002
#define MANCHESTER  0x0004
#define TONO_CRS    0x0008
#define NCRC_INS    0x0010
#define CRC_16      0x0020
#define BT_STUFF    0x0040
#define PAD         0x0080
#define CRSF        0x0000      /* no carrier sense filter */
#define CRS_SRC     0x0800
#define CDTF        0x0000      /* no collision detect filter */
#define CDT_SRC     0x8000
#define MIN_FRM_LEN 0x0040      /* 64 bytes */
#define MIN_DATA_LEN MIN_FRM_LEN - 18   /* assumes Ethernet/IEEE 802.3
                                           frames with 6 bytes of address */
#define MAX_FRAME_SIZE  1500 - 3
```

231421-21

```
/PCO/USR/CHUCK/CSRC/DLD.C


/************************************************************************
*                                                                       *
*                          82586 Handler                                *
*                                                                       *
************************************************************************/

/* Define constants for storage area */

#define CB_CNT      8   /* Number of available Command Blocks */
#define FD_CNT      16  /* Number of available Frame Descriptors */
#define RBD_CNT     64  /* Number of available Receive Buffer descriptors */
#define TBD_CNT     16  /* Number of available Transmit Buffer descriptors */

/* loopback parameters passed to Configure() */

#define INTERNAL_LOOPBACK   0x4000
#define EXTERNAL_LOOPBACK   0x8000
#define NO_LOOPBACK         0x0000

#include "dld.h"            /* 586 Data Structures */


/* 186 Timer Addresses */

#define TIMER1_CTL  0xFF5E
#define TIMER1_CNT  0xFF58
#define TIMER2_CTL  0xFF66
#define TIMER2_CNT  0xFF60


/* external functions */

/* I/O */
int     inw();      /* input word : inw(address) */
void    outw();     /* output word: outw(address, value) */
void    init_intv();/* initialize the interrupt vector table */
void    enable();   /* enable 80186 interrupts */
void    disable();  /* disable 80186 interrupts */

extern  char    *Build_Ptr();

u_short     SEGMT;      /* Data segment value */
char        *pNULL;     /* NULL pointer */

/* Macro 'type' of definitions */

#define CA  outw(0xCB,0)    /* the command to issue a Channel Attention */

#define ESI_LOOPBACK outw(0xCB,0) /* put the ESI in Loopback */
#define NO_ESI_LOOPBACK outw(0xCB,8) /* take the ESI out of Loopback */

#define EOI_80130         outb(0xE0,0x63)    /* End Of Interrupt */
#define TIMER1_EOI_80186 outw(0xFF22,0x04) /* EOI for Timer 1 on the 186 */
#define TIMER1_EOI_80130 outb(0xE0,0x64) /*EOI for 186's Timer1 on the 130 */
```

231421-22

```
unsigned long   good_xmit_cnt;
u_short         underrun_cnt;
u_short         no_crs_cnt;
unsigned long   defer_cnt;
u_short         sqe_err_cnt;
u_short         max_col_cnt;
unsigned long   recv_frame_cnt;
u_short         reset_cnt;

    /* Allocate storage for structures and buffers */


struct FLAGS flags;


/* 586 structures */

/* System Configuration Pointer: Rom Initialization */
/* struct SCP scp = {0x0000,0x0000,0x0000,0x1FF6,0x0000};  */

/* struct ISCP iscp;     Intermediate System Configuration Pointer */

struct SCB scb;      /* System Control Block */

struct CB cb[CB_CNT],    /* Command Blocks */
*cb_tos, *begin_cbl, *end_cbl;
                         /* pointer to the beginning of the free
                            command block list (cb_tos) and the
                            beginning and end of the 82586 cbl */

struct TBD tbd[TBD_CNT],    /* Transmit Buffer Descriptor */
*tbd_tos;                /* pointer to the free Transmit buffer
                            descriptors */

struct TB tbuf[TBD_CNT];     /* Transmit Buffers */


struct FD fd[FD_CNT],        /* Frame Descriptors */
*begin_fd, *end_fd;       /* pointers to the beginning and end of
                             the free FD list */

struct RBD rbd[RBD_CNT],     /* Receive Buffer Descriptors */
```

231421-23

```
/PCO/USR/CHUCK/CSRC/DLD.C


*begin_rbd, *end_rbd;          /* pointers to the beginning and the
                                  end of the rbd list */

struct RB rbuf[RBD_CNT];       /* Receive Buffers */

struct MAT mat[MULTI_ADDR_CNT];      /* Multicast Address Table */
struct MA_CB ma_cb;                  /* Multicast Address Command Block */


/* The following structures are used only in Reset_586() function */
struct      CB      res_cb;      /* Temporary CB for reinitializing the 586 */
struct      MA_CB   res_ma_cb;   /* Temporary MA_CB for reloading Multicast */

/*  Hardware Support Functions */

Enable_586_Int()
{
    int     c;

    c = inb(0xE2);          /* read the 80130 interrupt mask register */
    outb(0xE2, 0x00F7 & c); /* write to the 80130 interrupt mask register */
}

Disable_586_Int()
{
    int     c;

    c = inb(0xE2);
    outb(0xE2, 0x0008 | c);
}

Set_Timeout()
{
    outw(TIMER1_CNT, 0);    /* Write a 0 to Timer1 count register */
    outw(0xFF5E, 0xE009);   /* Set ENable bit in Timer1 Mode/Control register */
}

Reset_Timeout()
{
    outw(0xFF5E, 0x6009); /* Reset ENable bit in Timer1 Mode/Control register */
}

Init_Timer() /* 186's Timer 2 is a prescaler for Timer 1. It clocks Timer 1
                every 32.7 msec. The deadman timeout is set for 1.25 sec */
{
    outw(0xFF38, 0x000C);    /* Set Timer1 Interrupt Control register */
    outw(0xFF62, 0xFFFF);    /* set max count register for timer2 to OFFFFH */
    outw(0xFF5A,     38);    /* set max count register A for timer 1 */
    outw(0xFF66, 0xC001);    /* Set Timer2 Mode/Control register */
    outw(0xFF5E, 0x6009);    /* Set Timer1 Mode/Control register */
    outw(0xFF28, (inw(0xFF28) & 0xFFEF)); /* Enable 186 Timer1 interrupt */
    outb(0xE2,(inb(0xE2) & 0x00EF));  /* enable 80130 interrupt from 80186 */
}

/* end hardware support functions */

Clear_Cnt()
```

231421-24

```
/PCO/USR/CHUCK/CSRC/DLD.C

{
    scb.crc_errs = 0;        /* clear 586 error statistic counters */
    scb.aln_errs = 0;
    scb.rsc_errs = 0;
    scb.ovr_errs = 0;

    good_xmit_cnt = 0;       /* init data link statistics */
    underrun_cnt = 0;
    no_crs_cnt = 0;
    defer_cnt = 0;
    sqe_err_cnt = 0;
    max_col_cnt = 0;
    recv_frame_cnt = 0;
    reset_cnt = 0;
}

Init_586()
{
    struct ISCP *piscp;
    u_short i;
    struct  MAT *pmat;

    NO_ESI_LOOPBACK;  /* Done for 82501. Inactivates CRS if powered up
                         in loopback */
    ESI_LOOPBACK;

    init_intv();     /* Initialization DLDs interrupt vectors */

    Init_Timer();

    flags.reset_seme = 0;    /* Initialize Reset Flags */
    flags.reset_pend = 0;
    flags.stat_on = 1;

    Disable_586_Int();


    piscp =  0x0000FFF0 ; /* Initialize the ISCP pointer*/
    piscp->busy = 1;
    piscp->offset =  Offset(&scb);
    piscp->base1 = SEGMT << 4;
    piscp->base2 = (SEGMT >> 12) & 0x000F ;

    pNULL = Build_Ptr(NULL);     /* build a NULL pointer - 8086 type: 32 bits */
    Build_Rfa();     /* init Receive Frame Area */
    Build_Cb(); /* init Command Block list */
    ma_cb.cmd = 0;       /* multicast address semaphore init */

    Clear_Cnt();

    scb.stat = 0;

    CA;      /* wait for the 586 to complete initialization */

    for ( i = 0; i <= 0xFF00; i++)
```

231421-25

```
/PCO/USR/CHUCK/CSRC/DLD.C

        if (scb.stat == (CX | CNA))
            break;

    if (i >0xFF00)
    Fatal("DLD:init - Did not get an interrupt after Reset/CA\n");

    /* Ack the reset Interrupt */
    scb.cmd = (CX | CNA);
    CA;
    Wait_Scb();
    Enable_586_Int();

    scb.cbl_offset = Offset(&cb[0]);    /* link scb to cb and fd lists */
    scb.rfa_offset = Offset(&fd[0]);

     /* move the prom bytes into whoami array */

    for (i = 0; i < ADD_LEN; i++)
        whoami[(ADD_LEN - 1) - i] = inb(whoami_io_add + i*2);

        /* Initialization the Multicast Address Table */

    for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        pmat->stat = FREE;

    Configure(INTERNAL_LOOPBACK);    /* Put 586 in internal loopback */

    SetAddress();    /* Set up the station address */

    /* run diagnostics */

    Test_Link();

    if (Self_Test != PASSED)
        return(Self_Test);

    Configure(NO_LOOPBACK); /* Configure the 82586 */

    return(Self_Test);
}

Build_Rfa()
{
    struct   FD      *pfd;
    struct   RBD     *prbd;
    struct   RB      *pbuf;
    unsigned long    badd;

    /* Build a linear linked frame descriptor list */

    for (pfd = &fd[0]; pfd <= &fd[FD_CNT - 1]; pfd++) {

        pfd->stat = pfd->el_s = 0;
        pfd->link = Offset(pfd+1);
        pfd->rbd_offset = NULL;
    }
```

231421-26

/PCO/USR/CHUCK/CSRC/DLD.C

```
    end_fd = --pfd;             /* point to &fd[FD_CNT - 1] */
    pfd->link = NULL;           /* last fd link is NULL */
    pfd->el_s = ELBIT;          /* last fd has EL bit set */
    begin_fd = pfd = &fd[0];    /* point to first fd */
    pfd->rbd_offset = Offset(&rbd[0]);  /* link first fd to first rbd */

    /* Build a linear linked receive buffer descriptor list */

    for (prbd = &rbd[0], pbuf = &rbuf[0]; prbd <= &rbd[RBD_CNT - 1];
                                                    prbd++, pbuf++) {
        badd = SEGMT << 4;
        badd += Offset(pbuf);
        prbd->buff_l = badd;
        prbd->buff_h = badd >> 16;
        prbd->buff_ptr = pbuf;

        prbd->act_cnt =  0;
        prbd->link = Offset(prbd + 1);
        prbd->size = RBUF_SIZE;
    }

    end_rbd = --prbd;
    prbd->link = NULL;      /* last rbd points to NULL */
    prbd->size |= ELBIT;    /* last rbd has el bit set */

    begin_rbd = &rbd[0];
}

Build_Cb()  /* Build a stack of free command blocks */
{
    struct  CB  *pcb;
    struct  TBD *ptbd;
    struct  TB  *pbuf;
    unsigned    long    badd;

    for (pcb = &cb[0]; pcb <= &cb[CB_CNT - 1]; pcb++) {
        pcb->stat = 0;
        pcb->cmd = ELBIT;
        pcb->link = Offset(pcb + 1);
    }
    --pcb;
    begin_cbl = end_cbl = pNULL;
    pcb->link = NULL;
    cb_tos = &cb[0];

    /* Build a stack of transmit buffer descriptors */

    for (ptbd = &tbd[0], pbuf = &tbuf[0]; ptbd <= &tbd[TBD_CNT - 1];
                                                    ptbd++, pbuf++) {

        ptbd->act_cnt  = TBUF_SIZE;
        ptbd->link = Offset(ptbd + 1);

        badd = SEGMT << 4;
```

231421-27

/PCO/USR/CHUCK/CSRC/DLD.C

```
        badd += Offset(pbuf);
        ptbd->buff_l = badd;
        ptbd->buff_h = badd >> 16;
        ptbd->buff_ptr = pbuf;
    }

    --ptbd;
    ptbd->link = NULL;   /* last tbd link is NULL */
    tbd_tos = &tbd[0];   /* Set the Top Of the Stack */
}

    /* Get a Command Block from the free list */

struct  CB   *Get_Cb() /* return a pointer to a free command block */
{
    struct  CB *pcb;

    if (Offset(pcb = cb_tos) == NULL)
        return(pNULL);
    cb_tos = (struct CB *) Build_Ptr(pcb->link);
    pcb->link = NULL;
    return(pcb);
}

/* Put a Command Block back onto the free list */

Put_Cb(pcb)

    struct  CB *pcb;

{
    pcb->stat = 0;
    pcb->link = Offset(cb_tos);
    cb_tos = pcb;
}

struct TBD  *Get_Tbd()  /* return a pointer to a free transmit buffer
                           descriptor */
{
    struct   TBD    *ptbd;

    flags.reset_sema = 1;
    Disable_586_Int();
    if ((ptbd = tbd_tos) != pNULL) {
        tbd_tos = (struct TBD *) Build_Ptr(ptbd->link);
        ptbd->link = NULL;
    }
    Enable_586_Int();
    flags.reset_sema = 0;
    if (flags.reset_pend == 1)
        Reset_586();
    return(ptbd);
}

Put_Tbd(ptbd)
```

231421-28

/PCO/USR/CHUCK/CSRC/DLD.C

```
struct      TBD     *ptbd;
{
    struct      TBD     *p ;

    /* find the end of the tbd list returned, ptbd is the beginning */

    for (p = ptbd; p->link != NULL; p = (struct TBD *) Build_Ptr(p->link)) ;

    p->act_cnt = TBUF_SIZE;      /* clear EOFBIT and update size on last tbd */
    p->link = Offset(tbd_tos);
    tbd_tos = ptbd;
}


SetAddress()
{
    struct  CB  *pcb;

#ifdef DEBUG

    if ((pcb = Get_Cb()) == pNULL)
    Fatal("dld.c - SetAddress - couldn't get a CB\n");

#else

    pcb = Get_Cb();

#endif  /* DEBUG */

    bcopy((char *)&pcb->parm1, &whoami[0], ADD_LEN); /* move the prom
                                                        address to IA cmd */
    pcb->cmd = IA | ELBIT;

    Issue_CU_Cmd(pcb);
}

Wait_Scb()  /* wait for the scb command word to be clear */
{
    u_short     i, stat;

    for (stat = FALSE; stat == FALSE; ) {

        for (i=0; i<=0xFF00; i++)
            if (scb.cmd == 0)
                break;

        if (i > 0xFF00) {
            Bug("DLD: Scb command not clear\n");
            CA;
        }
        else
            stat = TRUE;
    }
```

231421-29

/PCO/USR/CHUCK/CSRC/DLD.C

```
}

Issue_CU_Cmd(pcb) /* Queue up a command and issue a start CU command if no
                     other commands are queued */
    struct  CB  *pcb;
{
    Disable_586_Int();
    if (begin_cbl == pNULL) {   /* if the list is inactive start CU */
        begin_cbl = end_cbl = pcb;
        scb.cbl_offset = Offset(pcb);
        Wait_Scb();
        scb.cmd = CU_START;
        Set_Timeout();              /* set deadman timer for CU */
        CA;
    }
    else {
        end_cbl->link = Offset(pcb);
        end_cbl = pcb;
    }
    Enable_586_Int();
}

Isr7()
{
    outb(0xE0, 0x67);    /* EOI 80130 */
}

Isr6()
{
    Write("\nInterrupt 6\n");
    outb(0xE0, 0x66);    /* EOI 80130 */
}

Isr5()
{
    Write("\nInterrupt 5\n");
    outb(0xE0, 0x65);    /* EOI 80130 */
}

    /* Deadman Timer Interrupt Service Routine */

Isr_Timeout()   /* Interrupt 4 */
{
    Reset_Timeout();
    if (flags.reset_sema == 1)
        flags.reset_pend = 1;
    else
        Reset_586();

    TIMER1_EOI_80186;
    TIMER1_EOI_80130;
}

    /* Interrupt 0 is Uart in UAP Module */
    /* Interrupt 2 is Timer in UAP Module */
```

231421-30

```
/PCO/USR/CHUCK/CSRC/DLD.C


Isr1()
{
    Write("\nInterrupt 1\n");
    outb(0xE0, 0x61);   /* EOI 80130 */
}

    /* 586 Interrupt service routine: Interrupt 3 */

Isr_586()
{

    u_short     stat_scb;
    struct  CB      *pcb;

    enable();   /* nesting only the uart interrupt */

    Wait_Scb();
    scb.cmd = (stat_scb = scb.stat) & (CX | CNA | FR | RNR);
    CA;

    if (stat_scb & (FR | RNR))
        Recv_Int_Processing();

    if (stat_scb & CNA) {  /* end of cb processing */

        Reset_Timeout();    /* clear deadman timer  */
        pcb = Build_Ptr(scb.cbl_offset);

#ifdef DEBUG

        if (begin_cbl == pNULL){
            Bug("DLD: begin_cbl == NULL in interrupt routine\n");
            return;
        }

        if ((pcb->stat & 0xC000) != 0x8000)
            Fatal("DLD: C bit not set or B bit set in interrupt routine\n");

#endif /* DEBUG */

        switch (pcb->cmd & CMD_MASK) {

        case TRANSMIT:

            if (flags.stat_on == 1) {/* if Transmit Statistics are collected do */

                /* if sqe bit = 0 and there were no collisions -> sqe error
                   this condition will occur on the first transmission if
                   there were no collisions, or if the previous transmit
                   command reached the max collision count, and the current
                   transmission had no collisions */

                if ((pcb->stat & (SQEMASK | MAXCOLMASK | COLLMASK)) == 0)
                    ++sqe_err_cnt;

                if (pcb->stat & DEFERMASK)
                    ++defer_cnt;
```

231421-31

/PCO/USR/CHUCK/CSRC/DLD.C

```
            if (pcb->stat & NOERRBIT)
                ++good_xmit_cnt;
            else {
                if (pcb->stat & NOCRSMASK)
                    ++no_crs_cnt;
                if (pcb->stat & UNDERRUNMASK)
                    ++underrun_cnt;
                if (pcb->stat & MAXCOLMASK)
                    ++max_col_cnt;
            }
        }
        if (pcb->parm1 != NULL)
            Put_Tbd(Build_Ptr(pcb->parm1));
        break;

    case DIAGNOSE:

        flags.diag_done = 1;
        if ((pcb->stat & NOERRBIT) == 0)
            Self_Test = FAILED_DIAGNOSE;
        break;

    default:
        ;
    }

    /* check to see if another command is queued */

    if (pcb->link == NULL)
        begin_cbl = pNULL;
    else { /* restart the CU and execute the next command on the cbl */

        begin_cbl = Build_Ptr(pcb->link);
        scb.cbl_offset = pcb->link;
        Wait_Scb();
        scb.cmd = CU_START;
        CA;
        Wait_Scb();
        Set_Timeout();      /* START deadman timer */
    }
    if ((pcb->cmd & CMD_MASK) == MC_SETUP)
        pcb->cmd = 0; /* clear MC_SETUP cmd word, this will implement a
                         lock semephore so that it won't be reused until
                         it is completed */
    else
        Put_Cb(pcb); /* Don't return MC_SETUP cmd block. It's not a
                        general purpose command block from free CB list */
}
disable();  /* disable cpu int so that the 586 isr will not nest */
EOI_80130;
}
```

231421-32

/PCD/USR/CHUCK/CSRC/DLD.C

```
Recv_Int_Processing()
{
    struct      FD      *pfd;  /* points to the Frame Descriptor */
    struct      RBD     *q,    /* points to the last rbd for the frame */
                        *prbd; /* points to the first rbd for the frame */

    for (pfd = begin_fd; pfd != pNULL; pfd = begin_fd)
        if (pfd->stat & CBIT) {
            begin_fd = (struct FD *) Build_Ptr(pfd->link);
            prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);
            if (prbd != pNULL) { /* check to see if a buffer is attached */

#ifdef DEBUG
                if (prbd != begin_rbd)
                    Fatal("DLD: prbd != begin_rbd in Recv_Int_Processing\n");
#endif /* DEBUG */
                for (q = prbd; (q->act_cnt & EOFBIT) != EOFBIT;
                                        q = (struct RBD *) Build_Ptr(q->link));

                begin_rbd = (struct RBD *) Build_Ptr(q->link);
                q->link = NULL;
            }
            if (pfd->stat & OUT_OF_RESOURCES)
                Put_Free_RFA(pfd);
            else {
                /* if the DLD is in a loopback test, check the frame recv */
                if (flags.lpbk_mode == 1)
                    Loopback_Check(pfd);
                else

            /* if it's a multicast address check to see if it's
                    in the multicast address table, if not discard the frame */

              if ( ((pfd->dest_addr[0] & 01) == 01) && (!Check_Multicast(pfd)))
                    Put_Free_RFA(pfd);
              else
                {   Recv_Frame(pfd);
                    ++recv_frame_cnt;
                }
            }
        }
        else {
            Ru_Start(); /* If RU has gone into no resources, restart it */
            break;
        }
}

Loopback_Check(pfd)          /* Called by Recv_Int_Processing; checks address
                                and data of potential loopback frame */
    struct  FD  *pfd;
{
    struct RBD  *prbd;
    struct RB   *pbuf;
```

231421-33

```
/PCO/USR/CHUCK/CSRC/DLD.C

    if ( bcmp((char *) &pfd->src_addr[0], &whoami[0], ADD_LEN) != 0 ) {
        Put_Free_RFA(pfd);
        return;
    }
    prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset); /* point to receive
                                                         buffer descriptor */
    pbuf = (struct RB *) prbd->buff_ptr; /* point to receive buffer */

    if ( bcmp((char *) pbuf, &lpbk_frame[0], LPBK_FRAME_SIZE) != 0) {
        Put_Free_RFA(pfd);
        return;
    }

    flags.lpbk_test = 1;  /* passed loopback test */
    Put_Free_RFA(pfd);
}

Check_Multicast(pfd)    /* returns true if multicast address is in MAT */
    struct      FD  *pfd;
{
    struct      MAT *pmat;

    for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( pmat->stat == INUSE &&
        (bcmp((char *) &pfd->dest_addr[0], &pmat->addr[0], ADD_LEN) == 0))
        break;

    if (pmat > &mat[MULTI_ADDR_CNT - 1])
        return(FALSE);
    return(TRUE);
}

    /* Test the Link function: executes Diagnose and Loopback tests */

Test_Link()
{
    Self_Test = PASSED;
    Diagnose();
    if (Self_Test == FAILED_DIAGNOSE)
        return;
    Ru_Start();     /* start up the RU for loopback tests */
    flags.lpbk_mode = 1; /* go into loopback mode */

    flags.lpbk_test = 0; /* set looback test to false */
    Send_Lpbk_Frame();  /* internal loopback test */
    if (flags.lpbk_test == 0) {
        Self_Test = FAILED_LPBK_INTERNAL;
        flags.lpbk_mode = 0;
        return;
    }

    flags.lpbk_test = 0;
    Configure(EXTERNAL_LOOPBACK);   /* external loopback test w/ ESI in lpbk */
    Send_Lpbk_Frame();
    if (flags.lpbk_test == 0) {
        Self_Test = FAILED_LPBK_EXTERNAL;
```

231421-34

/PCO/USR/CHUCK/CSRC/DLD.C

```
        flags.lpbk_mode = 0;
        return;
    }

    flags.lpbk_test = 0;     /* external loopback test through transceiver */
    NO_ESI_LOOPBACK;
    Send_Lpbk_Frame();
    if (flags.lpbk_test == 0)
        Self_Test = FAILED_LPBK_TRANSCEIVER;

    flags.lpbk_mode = 0; /* leave loopback mode */
}

Send_Lpbk_Frame()
{
    struct   TBD    *ptbd;
    int             i;

    for (i = 0; i < 8; i++) {   /* send lpbk frame 8 times, since it's
                                   best effort delivery */

#ifdef DEBUG
        if ((ptbd = Get_Tbd()) == pNULL)
            Fatal("dld - Send_Lpbk_Frame - couldn't get a TBD\n");
#else
        ptbd = Get_Tbd();

#endif /* DEBUG */

        ptbd->act_cnt = EOFBIT ! LPBK_FRAME_SIZE;
        bcopy((char *) ptbd->buff_ptr, &lpbk_frame[0], LPBK_FRAME_SIZE);

        while(!Send_Frame(ptbd, &whoami[0]));
    }
}


Diagnose()
{
    struct    CB    *pcb;

#ifdef DEBUG
        if ((pcb = Get_Cb()) == pNULL)
        Fatal("dld - Diagnose - couldn't get a CB\n");
#else
        pcb = Get_Cb();

#endif /* DEBUG */

        flags.diag_done = 0;
        Self_Test = FALSE;
        pcb->cmd = DIAGNOSE ! ELBIT;

        Issue_CU_Cmd(pcb);

        while (flags.diag_done == 0) ;  /* wait for Diag cmd to finish */
```

231421-35

/PCD/UBR/CHUCK/CSRC/DLD.C

```
}

Configure(loopflag)

    u_short loopflag;
{
    struct  CB    *pcb;

#ifdef DEBUG
    if ((pcb = Get_Cb()) == pNULL)
        Fatal("dld - Configure - couldn't get a CB\n");
#else
    pcb = Get_Cb();

#endif /* DEBUG */


    /* Ethernet default parameters */

    pcb->parm1 = 0x080C;
    pcb->parm2 = 0x2600 | loopflag;
    pcb->parm3 = 0x6000;
    pcb->parm4 = 0xF200;
    pcb->parm5 = 0x0000;
    if (loopflag == NO_LOOPBACK)
        pcb->parm6 = 0x0040;
    else
        pcb->parm6 = 0x0006;     /* loopback frame is less bytes than
                                    the minimum frame length */
    pcb->cmd    = CONFIGURE | ELBIT;


    Issue_CU_Cmd(pcb);
}

/* Send a frame to the cable.  pass a pointer to the destination address
   and a pointer to the first transmit buffer descriptor.  */

Send_Frame(ptbd, padd)  /* returns false if it can't get a Command block */
struct      TBD     *ptbd;
char                *padd;
{
    struct      CB  *pcb;

    u_short         length;

    flags.reset_sema = 1;

    if ((pcb = Get_Cb()) == pNULL) {
        flags.reset_sema = 0;
        if (flags.reset_pend == 1)
            Reset_586();
        return(FALSE);
    }

    pcb->parm1 = Offset(ptbd);
```

231421-36

```
/PCO/USR/CHUCK/CSRC/DLD.C

    /* move destination address to command block */
    bcopy((char *)&pcb->parm2, (char *)padd, ADD_LEN);


    /* calculate the length field by summing up all the buffers */

    for (length = 0; ptbd->link != NULL; ptbd = Build_Ptr(ptbd->link))
        length += ptbd->act_cnt;

    length += (ptbd->act_cnt & 0x3FFF);  /* add the last buffer */

    /* check to see if padding is required, do not do padding on loopback */

    /* this will not work if MIN_DATA_LEN > TBUF_SIZE */

    if ((length < MIN_DATA_LEN) &&  /* assumes a 4 byte CRC */
        (bcmp(&whoami[0], (char *)padd, ADD_LEN) != 0))

            ptbd->act_cnt = MIN_DATA_LEN | EOFBIT;

    pcb->parm5 = length;        /* length field */


    pcb->cmd = TRANSMIT | ELBIT;


    Issue_CU_Cmd(pcb);
    flags.reset_sema = 0;
    if (flags.reset_pend == 1)
        Reset_586();
    return(TRUE);
}


Add_Multicast_Address(pma)  /* pma - pointer to multicast address */
char    *pma;               /* returning false means the Multicast address
                               table is full */
{
    struct      MAT     *pmat;

    flags.reset_sema = 1;

/* if the multicast address is a duplicate of one already in the MAT,
   then return */

    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( pmat->stat == INUSE &&
            (bcmp( &pmat->addr[0], (char *) pma, ADD_LEN) == 0)) {
            return(TRUE);
        }


    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if (pmat->stat == FREE) {
            pmat->stat = INUSE;
            bcopy( &pmat->addr[0], (char *) pma, ADD_LEN);
            break;
```

231421-37

/PCO/USR/CHUCK/CSRC/DLD.C

```
        }
    if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
        flags.reset_sema = 0;
        if (flags.reset_pend == 1)
            Reset_586();
        return(FALSE);
    }

    Set_Multicast_Address();
    flags.reset_sema = 0;
    if (flags.reset_pend == 1)
        Reset_586();
    return(TRUE);
}

Delete_Multicast_Address(pma)  /* returning false means the multicast address
                                  was not found */
char    *pma;
{
    struct      MAT     *pmat;

    flags.reset_sema = 1;

    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( pmat->stat == INUSE &&
            (bcmp( &pmat->addr[0], (char *) pma, ADD_LEN) == 0)) {
            pmat->stat = FREE;
            break;
        }

    if (pmat > &mat[MULTI_ADDR_CNT - 1]) {
        flags.reset_sema = 0;
        if (flags.reset_pend == 1)
            Reset_586();
        return(FALSE);
    }

    Set_Multicast_Address();
    flags.reset_sema = 0;
    if (flags.reset_pend == 1)
        Reset_586();
    return(TRUE);
}

Set_Multicast_Address()
{
    struct      MAT     *pmat;
    struct      MA_CB   *pma_cb;
    u_short     i;

    i = 0;
    pma_cb = &ma_cb;
    while (pma_cb->cmd != 0) ; /* if the MA_CB is inuse, wait until it's free */
    pma_cb->link = NULL;
```

231421-38

```
/PCD/USR/CHUCK/CSRC/DLD.C

    for (pmat = mat; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( pmat->stat == INUSE) {
            bcopy( &pma_cb->mc_addr[i], &pmat->addr[0], ADD_LEN);
            i += ADD_LEN;
        }

    pma_cb->mc_cnt = i;
    pma_cb->cmd = MC_SETUP | ELBIT;

    Issue_CU_Cmd(pma_cb);

}


Put_Free_RFA(pfd)  /* Return Frame Descriptor and Receive Buffer
                      Descriptors to the Free Receive Frame Area */

    struct      FD      *pfd;
{
    struct      RBD     *prbd,  /* points to beginning of returned RBD list */
                        *q;     /* points to end of returned RBD list */
    char        ru_start_flag_fd,  /* indicates whether to restart RU */
                ru_start_flag_rbd;

    flags.reset_sema = 1;
    ru_start_flag_fd = ru_start_flag_rbd = FALSE;
    pfd->el_s = ELBIT;
    pfd->stat = 0;
    prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);/* pick up the link to the rbd */
    pfd->link = pfd->rbd_offset = NULL;

    /*  Disable_586_Int(); this command is only necessary in a multitasking
    program. However in this single task environment this routine is originally
    called from isr_586(), therefore interrupts are already disabled */


    if (begin_fd == pNULL)
        begin_fd = end_fd = pfd;
    else {
        end_fd->link = Offset(pfd);
        end_fd->el_s = 0;
        end_fd = pfd;
        ru_start_flag_fd = TRUE;
    }

    if (prbd != pNULL) {        /* if there is a rbd attached to the fd then
                                   find the beginning and end of the rbd list */

        for (q = prbd; q->link != NULL; q = Build_Ptr(q->link))
            q->act_cnt = 0;

            /* now prbd points to the beginning of the rbd list and
               q points to the end of the list */

        q->size = RBUF_SIZE | ELBIT;
        q->act_cnt = 0;
```

231421-39

/PCO/USR/CHUCK/CSRC/DLD.C

```
        if (begin_rbd == pNULL) { /* if there is nothing on the list
                                     create a new list */
            begin_rbd = prbd;
            end_rbd = q;
            if (prbd != q)
                ru_start_flag_rbd = TRUE; /* if there is more than one rbd
                                             returned start the RU */
        }
        else {
                                /* if the rbd list already exists add on
                                   the new returned rbds */
            end_rbd->link = Offset(prbd);
            end_rbd->size = RBUF_SIZE;
            end_rbd = q;
            ru_start_flag_rbd = TRUE;
        }
    }
    if (ru_start_flag_fd && ru_start_flag_rbd)
        Ru_Start();

    /*  Enable_586_Int(); if Disable_586_Int() is used above */

    flags.reset_sema = 0;
    if (flags.reset_pend == 1)
        Reset_586();
}

Ru_Start()
{
    if ((scb.stat & RU_MASK) == RU_READY) /* if the RU is already 'ready'
                                             then return */
        return;

    if ((begin_fd->stat & CBIT) == CBIT)
        return;

    begin_fd->rbd_offset = Offset(begin_rbd); /* link the beginning of the rbd
                                                 list to the first fd */
    scb.rfa_offset = Offset(begin_fd);
    Wait_Scb();
    scb.cmd = RU_START;
    CA;
}

Software_Reset()
{
    scb.cmd = RESET;
    CA;
    Wait_Scb();
}

Issue_Reset_Cmds()
{
    Wait_Scb();
    scb.cmd = CU_START;
    CA;
```

231421-40

```
}

/* Execute a reset, Configure, SetAddress, and MC_Setup, then restart the
   Receive Unit and the Command Unit */
Reset_586()
{
    struct      MAT     *pmat;
    u_short     i;

    ++reset_cnt;
    Disable_586_Int();
    ESI_LOOPBACK;
    Software_Reset();

    scb.stat = 0;

    CA;     /* wait for the 586 to complete initialization */

    for ( i = 0; i <= 0xFF00; i++)
        if (scb.stat == (CX | CNA))
            break;

    if (i >0xFF00)
        Fatal("DLD:init - Did not get an interrupt after Software Reset\n");

    /* Ack the reset Interrupt */
    Wait_Scb();
    scb.cmd = (CX | CNA);
    CA;
    Wait_Scb();

#ifdef  DEBUG
    if ( begin_cbl == pNULL)
        Fatal("DLD: begin_cbl = NULL in Reset_586");
#endif  /* DEBUG */
```

231421-41

/PCO/USR/CHUCK/CSRC/DLD.C

```
    /* Configure the 586 */
    /* Ethernet default parameters; Configure is not necessary when using
       default parameters  */

    res_cb.link = NULL;

    res_cb.parm1 = 0x080C;
    res_cb.parm2 = 0x2600;
    res_cb.parm3 = 0x6000;
    res_cb.parm4 = 0xF200;
    res_cb.parm5 = 0x0000;
    res_cb.parm6 = 0x0040;
    res_cb.cmd   = CONFIGURE | ELBIT;

    scb.cbl_offset = Offset(&res_cb.stat);

    Issue_Reset_Cmds();

    /* Set the Individual Address */

    bcopy((char *) &res_cb.parm1, &whoami[0], ADD_LEN); /* move the prom
                                                           address to IA cmd */
    res_cb.cmd = IA | ELBIT;

    Issue_Reset_Cmds();

    /* reload the multicast addresses */

    i = res_ma_cb.stat = 0;
    res_ma_cb.link = NULL;

    for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
        if ( pmat->stat == INUSE ) {
            bcopy( &res_ma_cb.mc_addr[i], &pmat->addr[0], ADD_LEN);
            i += ADD_LEN;
        }

    res_ma_cb.mc_cnt = i;
    res_ma_cb.cmd = MC_SETUP | ELBIT;
    scb.cbl_offset = Offset(&res_ma_cb.stat);


    Issue_Reset_Cmds();

    /* Restart the Command Unit and the Receive Unit */

    flags.reset_sema = 0;
    flags.reset_pend = 0;

    NO_ESI_LOOPBACK;

    Recv_Int_Processing();

    scb.cbl_offset = begin_cbl;
    Wait_Scb();
```

231421-42

```
/PCO/USR/CHUCK/CSRC/DLD.C

    scb.cmd = CU_START;
    Set_Timeout();          /* Set Deadman Timer */
    CA;
    Enable_586_Int();
}


/* bcopy -- byte copy routine */
bcopy(dst, src, nbytes)
char    *dst, *src;
int     nbytes;
{
    while (nbytes--) *dst++ = *src++;
}

/* bcmp -- byte compare */
bcmp(s1, s2, nbytes)
char    *s1, *s2;
int     nbytes;
{
    while (nbytes-- && *s1++ == *s2++);
    return(*--s1 - *--s2);
}
```

231421-43

```
/PCO/USR/CHUCK/CSRC/LLC.C

/*************************************************************************
*                                                                        *
*              IEEE 802.2 Logical Link Control Layer                     *
*                     (Station Component)                                *
*************************************************************************/

#include "dld.h"

extern  char    *pNULL;

extern  struct  TBD *Get_Tbd();
extern  char        *Build_Ptr();

readonly char   xid_frame[XID_LENGTH] = { 0, 0, XID, 0xB1, 0x01, 0};
/* DSAP, SSAP, XID, xid class 1 response */

struct LAT lat[DSAP_CNT];

Init_Llc()
{
    struct  LAT     *plat;

    for (plat = &lat[0]; plat <= &lat[DSAP_CNT - 1]; plat++)
        plat->stat = FREE;
    return(Init_586());
}

/* Function for adding a new DSAP */

Add_Dsap_Address(dsap, pfunc) /* DSAP must be divisible by 2**(8-N), where
                                 2**N = DSAP_CNT.  (i.e. N LSBs must be 0).
                                 The function will return FALSE if does not
                                 meet the above requirements, or the Lsap
                                 Address Table is full, or the address has
                                 already been used.  NULL DSAP address is
                                 reserved for the Station Component */
int dsap, (*pfunc) ();
{
    struct  LAT     *plat;

    if ((dsap << (8-DSAP_SHIFT) & 0x00FF) != 0 || dsap == 0)
        return (FALSE);

    /* Check for duplicate dsaps.  */
    if ( (plat = &lat[dsap >> DSAP_SHIFT])->stat == FREE) {
            plat->stat = INUSE;
            plat->p_sap_func = pfunc;
            return (TRUE);
    }
    else
        return(FALSE);
}

/* Function for deleting DSAPs */

Delete_Dsap_Address(dsap) /* If the specified connection exists, it is severed.
                             If the connection does not exist, the command is ignored.  */
```

231421-44

```
        if (pfs->dsap == 0) {   /* if the frame is addressed to the Station
                                   Component, then a response may be required */

            if ( !(pfs->ssap & C_R_BIT) ) {/* if the frame received is a response,
                                              instead of a command, then reject it.
                                              Because this software does not implement
                                              DUPLICATE_ADDRESS_CHECK. -> no response
                                              frames should be recv'd */
                Station_Component_Response(pfd);
            }
        }
        /* not addressed to Station Component. */
        /* check to see if the dsap addressed is active */
        else if ((pfs->dsap << (8-DSAP_SHIFT) & 0x00FF) == 0 &&
                 (plat = &latE[(pfs->dsap) >> DSAP_SHIFT])->stat == INUSE ) {
            (*plat->p_sap_func)(pfd);   /* call the function associated
                                           with the dsap received */
            return;
        }
    }
    Put_Free_RFA(pfd); /* return the pfd if not given to the user saps */
}


Station_Component_Response(pfd)

    struct  FD      *pfd;
{
    struct FRAME_STRUCT     *prfs, *ptfs;
    struct TBD              *ptbd, *begin_ptbd, *q;
    struct RBD              *prbd;

    prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset);
    prfs  = (struct FRAME_STRUCT *) prbd->buff_ptr;

    switch (prfs->cmd & ~P_F_BIT)
        {
        case    XID:
```

231421-45

/PCO/USR/CHUCK/CSRC/LLC.C

```
            while ((ptbd = Get_Tbd()) == pNULL);
            ptbd->act_cnt = EOFBIT | XID_LENGTH;
            bcopy ((char *) ptbd->buff_ptr, &xid_frame[0], XID_LENGTH);
            ptfs = (struct FRAME_STRUCT *) ptbd->buff_ptr;
            ptfs->cmd = prfs->cmd;

            ptfs->dsap = prfs->ssap | C_R_BIT; /* return the frame
                                                  to the sender */
            ptfs->ssap = 0;
            while(!Send_Frame(ptbd, Build_Ptr(pfd->src_addr)));
            break;

        case    TEST:

            for (prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset),
                        q = begin_ptbd = pNULL; prbd != pNULL;
                                prbd = Build_Ptr(prbd->link)) {

                while ((ptbd = Get_Tbd()) == pNULL);
                if (q != pNULL)
                    q->link = Offset(ptbd);
                else
                    begin_ptbd = ptbd;
                ptbd->act_cnt = prbd->act_cnt;
                bcopy((char *) ptbd->buff_ptr, (char *) prbd->buff_ptr,
                                                  ptbd->act_cnt & 0x3FFF);
                q = ptbd;
            }

            ptfs = (struct FRAME_STRUCT *) begin_ptbd->buff_ptr;
            ptfs->cmd = prfs->cmd;

            ptfs->dsap = prfs->ssap | C_R_BIT;  /* return the frame to
                                                   the sender */
            ptfs->ssap = 0;
            while(!Send_Frame(begin_ptbd, Build_Ptr(pfd->src_addr)));
            break;

        }
}
```

231421-46

```
/PCO/USR/CHUCK/CSRC/UAP.C

/*************************************************************************
*                                                                        *
*                      User Application Program                          *
*               Async to IEEE 802.2/802.3 Protocol Converter             *
*                                                                        *
*************************************************************************/

#include "dld.h"

/* ASCII Characters */
#define ESC       0x1B
#define LF        0x0A
#define CR        0x0D
#define BS        0x08
#define BEL       0x07
#define SP        0x20
#define DEL       0x7F
#define CTL_C         0x03

/* Hardware */
#define CH_B_CTL     0x00DE
#define CH_A_CTL     0x00DC
#define CH_B_DAT     0x00DA
#define CH_A_DAT     0x00D8
#define UART_STAT_MSK    0x70

/* Interrupt cases for 8274 */
#define UART_TX_B        0
#define UART_RECV_B      0x08
#define UART_RECV_ERR_B 0x0C
#define  EXT_STAT_INT_B 0x04
#define  EXT_STAT_INT_A 0x14

char     fifo_t[256];
char     fifo_r[256];
char     wra[5], wrb[5];
unsigned     char     in_fifo_t, out_fifo_t, in_fifo_r, out_fifo_r, actual;
u_short      t_buf_stat, r_buf_stat;

char     cbuf[80];    /* Command line buffer */
char     line[81];    /* Monitor Mode display line */

unsigned     char     dsap, ssap, send_flag, local_echo;
char     Dest_Addr[ADD_LEN];
char     Multi_Addr[ADD_LEN];

int tmstat; /* terminal mode status: for leaving terminal mode */
int dhex, monitor_flag, hs_stat;        /* flags */


extern  struct TBD       *Get_Tbd();
extern  char             *Build_Ptr();

extern  struct FLAGS     flags;

extern  char     xid_frame[];
extern  char     whoami[];
```

231421-47

```
/PCD/USR/CHUCK/CSRC/UAP.C

extern  struct  MAT     mat[];
extern  struct  LAT     lat[];
extern  char    *pNULL;

extern  unsigned long   good_xmit_cnt;
extern  u_short         underrun_cnt;
extern  u_short         no_crs_cnt;
extern  unsigned long   defer_cnt;
extern  u_short         sqe_err_cnt;
extern  u_short         max_col_cnt;
extern  unsigned long   recv_frame_cnt;
extern  u_short         reset_cnt;

extern  struct  SCB     scb;

/* Macro 'type' of definitions */

#define RTS_ONB outb(CH_B_CTL,0x05);outb(CH_B_CTL,wrb[5]=wrb[5]|0x02)
#define RTS_OFFB outb(CH_B_CTL,0x05);outb(CH_B_CTL,wrb[5]=wrb[5]&0xFD)
#define RTS_ONA outb(CH_A_CTL,0x05);outb(CH_A_CTL,wra[5]=wra[5]|0x02)
#define RTS_OFFA outb(CH_A_CTL,0x05);outb(CH_A_CTL,wra[5]=wra[5]&0xFD)
#define UART_TX_DI_B outb(CH_B_CTL,0x01);outb(CH_B_CTL,wrb[1]=wrb[1]&0xFD)
#define UART_TX_EI_B outb(CH_B_CTL,0x01);outb(CH_B_CTL,wrb[1]=wrb[1]|0x02)
#define UART_RX_DI_B outb(CH_B_CTL,0x01);outb(CH_B_CTL,wrb[1]=wrb[1]&0xE7)
#define UART_RX_EI_B outb(CH_B_CTL,0x01);outb(CH_B_CTL,wrb[1]=wrb[1]|0x10)
#define RESET_TX_INT outb(CH_B_CTL,0x28)
#define EOI_8274 outb(CH_A_CTL,0x38) /* 8274 int is IR3 on 80130 */
#define EOI_80130_8274 outb(0xE0,0x60)
#define EOI_80130_TIMER outb(0xE0,0x62)


Enable_Uart_Int()
{
    int     c;

    c = inb(0xE2);  /* read the 80130 interrupt mask register */
    outb(0xE2, 0x00FE & c); /* write to the 80130 interrupt mask register */
}

Disable_Uart_Int()
{
    int     c;

    c = inb(0xE2);
    outb(0xE2, 0x0001 | c);
}

Enable_Timer_Int()
{
    int     c;

    outb(0xEA, 125);
    outb(0xEA, 0x00); /* Timer 1 interrupts every .125 sec */
    send_flag = FALSE;
    c = inb(0xE2);  /* read the 80130 interrupt mask register */
    outb(0xE2, 0x00FB & c); /* write to the 80130 interrupt mask register */
}
```

231421-48

/PCO/USR/CHUCK/CSRC/UAP.C

```
}

Read_Char()
{
    unsigned char   i;

    Read(&cbuf[0], 80, &actual);
    i = Skip(&cbuf[0]);
    return(cbuf[i]);
}

Write(pmsg)
    char    *pmsg;
{
    while (*pmsg != '\0') {
        if (*pmsg == '\n')
            Co(CR);
        Co(*pmsg++);
    }
}

Fatal(pmsg) /* write a message to the screen then stop */
    char    *pmsg;
{
    Write("Fatal: ");
    Write(pmsg);
    for(;;);
}

Bug(pmsg)   /* write a message to the screen then continue */
    char    *pmsg;
{
    Write("Bug: ");
    Write(pmsg);
}

Ascii_To_Char(c) /* convert ASCII-Hex to Char */
    char    c;
{
    if (('0' <= c) && (c <= '9'))
        return(c - '0');
    if (('A' <= c) && (c <= 'F'))
        return(c - 0x37);
    if (('a' <= c) && (c <= 'f'))
        return(c - 0x57);
    return(0xFF);
}

Lower_Case(c)
char    c;
{
    if (('a' <= c) && (c <= 'z'))
        return(c);
    if (('A' <= c) && (c <= 'Z'))
        return(c + 0x20);
    return(0);
}
```

231421-50

1

/PCO/USR/CHUCK/CSRC/UAP.C

```
Char_To_Ascii(c, ch) /* convert char to ASCII-Hex */
    unsigned char   c, ch[];
{
    unsigned char   i;

    i = (c & 0xF0) >> 4;
    if (i < 10)
        ch[0] = i + 0x30;
    else
        ch[0] = i + 0x37;
    i = (c & 0x0F) ;
    if (i < 10)
        ch[1] = i + 0x30;
    else
        ch[1] = i + 0x37;
    ch[2] = '\0';
}

Skip(pmsg)  /* skip blanks */
    char    *pmsg;
{
    int     i;

    for (i = 0; *pmsg == ' '; i++, pmsg++);
    return(i);
}

Read_Int()  /* Read a 16 bit Integer */
{
    u_short     wd, wh, wd1, wh1, j;
    char        i, done, hex, dover, hover;

    for (done = FALSE; done == FALSE; ) {
        Read(&cbuf[0], 80, &actual);
        i = Skip(&cbuf[0]);

        for (hex = dover = hover = FALSE, wd = wh = wd1 = wh1 = 0;
                        (j = Ascii_To_Char(cbuf[i])) <= 15; i++) {
            if (j > 9)
                hex = TRUE;
            wd = wd*10 + j;
            wh = wh*16 + j;
            if (wd < wd1)
                dover = TRUE;
            if (wh < wh1)
                hover = TRUE;
            wd1 = wd; wh1 = wh;
        }
        if (cbuf[i] == 'H' || cbuf[i] == 'h' || cbuf[i] == CR ||
                                cbuf[i] == LF || cbuf[i] == ' ') {
            if (cbuf[i] == 'H' || cbuf[i] == 'h')
                hex = TRUE;
            if (hex == TRUE && hover == FALSE)
                done = TRUE;
            if (hex == FALSE && dover == FALSE)
                done = TRUE;
```

231421-51

/PCO/USR/CHUCK/CSRC/UAP.C

```
            if (!done) {
                Write("\n This number is too big.\n It has to be less than 65536.\n");
                Write("\n Enter number --> ");
            }
        }
        else
            Write(" Illegal Character\n Enter a number -->");
    }
    if (hex)
        return(wh);
    return(wd);
}

Int_To_Ascii(value, base, ld, ch, width) /* convert an integer to an ASCII string */
    unsigned long   value;
    u_short      base, width;
    char    ch[], ld;
{
    u_short i, j;

    for (i = 0; i < width; i++) {
        j = value % base;
        if (j < 10)   ch[i] = j + 0x30;
        else ch[i] = j + 0x37;
        value = value / base;
    }
    for (i = width - 1; ch[i] == '0' && i > 0; i--)
        ch[i] = ld;
    ch[width] = '\0';
}

Write_Long_Int(dw, i)
    unsigned long   dw;
    u_short     i;
{
    u_short     j;
    char    ch[11];

    if (dhex)
        Int_To_Ascii(dw, 16, ' ', &ch[0], 8);
    else
        Int_To_Ascii(dw, 10, ' ', &ch[0], 10);
    for (j = 0; ch[j] != '\0'; i--, j++)
        line[i] = ch[j];
}

Write_Short_Int(w, i)
    u_short w, i;
{
    u_short j;
    char    ch[6];
    unsigned long   dw;

    dw = w;
    if (dhex)
        Int_To_Ascii(dw, 16, '0', &ch[0], 4);
    else
```

231421-52

/PCO/USR/CHUCK/CSRC/UAP.C

```
        Int_To_Ascii(dw, 10, '0', &ch[0], 5);
    for (j = 0; ch[j] != '\0'; i--, j++)
        line[i] = ch[j];
}


Yes()
{
    char    b;

    for ( ; ; ) {
        b = Read_Char();
        if ((b == 'Y') || (b == 'y'))
            return(TRUE);
        if ((b == 'N') || (b == 'n'))
            return(FALSE);
        Write(" Enter a Y or N --> ");
    }
}

Read_Addr(pmsg, add, cnt) /* pmsg - pointer to the output message */
                          /* add - pointer to the address */
                          /* cnt - number of bytes in the address */
    char    *pmsg, add[], cnt;
{
    char    i, j;

    for ( ; ; ) {
        Write(pmsg);
        Read(&cbuf[0], 80, &actual);
        for (j = skip(&cbuf[0]), i = 0; i < 2*cnt ; i++, j++) {
            if (('0' <= cbuf[j]) && (cbuf[j] <= '9'))
                cbuf[i] = cbuf[j] - '0';
            else
                if (('A' <= cbuf[j]) && (cbuf[j] <= 'F'))
                    cbuf[i] = cbuf[j] - 0x37;
            else
                if (('a' <= cbuf[j]) && (cbuf[j] <= 'f'))
                    cbuf[i] = cbuf[j] - 0x57;
            else {
                Write(" Illegal Character\n");
                break;
            }
        }
        if (i >= 2*cnt - 1)
            break;
    }
    for (i = 0; i <= cnt - 1; i++)
        add[(cnt - 1) - i] = cbuf[2*i] << 4 | cbuf[2*i + 1];
}

Write_Addr(padd, cnt)
    char    padd[], cnt;
{
    unsigned char   i, c[3];

    for ( ; cnt >0 ; cnt--) {
```

231421-53

/PCO/USR/CHUCK/CSRC/UAP.C

```
            break;

        case    TEST:

            for (prbd = (struct RBD *) Build_Ptr(pfd->rbd_offset),
                        q = begin_ptbd = pNULL; prbd != pNULL;
                                prbd = Build_Ptr(prbd->link)) {
                while ((ptbd = Get_Tbd()) == pNULL);
                if (q != pNULL)
                    q->link = Offset(ptbd);
                else
                    begin_ptbd = ptbd;
                ptbd->act_cnt = prbd->act_cnt;
                bcopy((char *) ptbd->buff_ptr, (char *) prbd->buff_ptr,
                                                    ptbd->act_cnt & 0x3FFF);
                q = ptbd;
            }

            ptfs = (struct FRAME_STRUCT *) begin_ptbd->buff_ptr;
            ptfs->cmd = prfs->cmd;

            ptfs->dsap = prfs->ssap | C_R_BIT;  /* return the frame to
                                                   the sender */
            ptfs->ssap = ssap;
            while(!Send_Frame(begin_ptbd, Build_Ptr(pfd->src_addr)));
            break;

    }
    Put_Free_RFA(pfd);  /* return the frame */
}

Fifo_T_Out()    /* called by main program */
{
    char    c;

    c = fifo_t[out_fifo_t++];

    Disable_Uart_Int();
    if (out_fifo_t == in_fifo_t)    /* if the fifo is empty */
        t_buf_stat = EMPTY; /* stop filling Transmit Buffer Descriptors */
    else        /* if the fifo was full and is now draining */
        if (t_buf_stat == FULL && out_fifo_t - 80 == in_fifo_t) { /* turn on
                                                                    the spigot */
            RTS_ONB;
            t_buf_stat = INUSE;
        }
    Enable_Uart_Int();
    return(c);
}

Fifo_T_In(c)    /* called by Uart receive interrupt */
    char    c;
{

    fifo_t[in_fifo_t++] = c;
    if (t_buf_stat == EMPTY)
```

231421-55

```
/PCO/USR/CHUCK/CSRC/UAP.C

        t_buf_stat = INUSE; /* start filling Transmit Buffer Descriptor */
    else        /* if there are only 20 locations left, turn off the spigot */
        if (t_buf_stat == INUSE && in_fifo_t + 20 == out_fifo_t) {
            RTS_OFFB;
            t_buf_stat = FULL;
        }
}

Fifo_R_Out()    /* called by transmit interrupt */
{
    char    c;

    c = fifo_r[out_fifo_r++];

    if (out_fifo_r == in_fifo_r)    /* if the fifo is empty */
        r_buf_stat = EMPTY;
    else        /* if the fifo was full and is now draining */
        if (r_buf_stat == FULL && out_fifo_r - 81 == in_fifo_r)
            r_buf_stat = INUSE;
    return(c);
}

Fifo_R_In(c)    /* called by Recv_Data_1() */
    char    c;
{
    fifo_r[in_fifo_r++] = c;
    Disable_Uart_Int();
    if (r_buf_stat == EMPTY) {
        UART_TX_EI_B;
        Co(0);              /* prime the interrupt */
        r_buf_stat = INUSE;
    }
    else        /* if the buffer is full, indicate it */
        if (r_buf_stat == INUSE && in_fifo_r == out_fifo_r)
            r_buf_stat = FULL;
    Enable_Uart_Int();
}

Isr_Uart()
{
    int     stat;
    char    c;

    outb(CH_B_CTL, 2);  /* point to RR2 in 8274 */

    switch(inb(CH_B_CTL) & 0x1C){ /* read 8274 interrupt vector and service it */

    case UART_TX_B:

        if (r_buf_stat == EMPTY) {
            UART_TX_DI_B;           /* if fifo is empty disable transmitter */
            RESET_TX_INT;
        }
        else
            outb (CH_B_DAT, Fifo_R_Out());
        break;
```

231421-56

```
/PCO/USR/CHUCK/CBRC/UAP.C

    case UART_RECV_ERR_B:

        outb(CH_B_CTL, 1); /* point to RR1 in 8274 */
        stat = inb(CH_B_CTL);
        outb(CH_B_CTL, 0x30);
        if (stat & 0x0010)
            Write("\nParity Error Detected\n");
        if (stat & 0x0020)
            Write("\nOverrun Error Detected\n");
        if (stat & 0x0040)
            Write("\nFraming Error Detected\n");
        break;

    case UART_RECV_B:

        c = inb(CH_B_DAT);

        if (hs_stat == TRUE) {
            hs_stat = FALSE;  /* Flag to terminate High Speed Transmit mode */
            break;
        }

        if (local_echo)
            Co(c);          /* echo the char back to the terminal; could cause
                               a transmit overrun if Tx interrupt is enabled */

        if (c == CTL_C)
            tmstat = FALSE;
        else
            Fifo_T_In(c);
        break;

    case EXT_STAT_INT_B:

        outb(CH_B_CTL, 0x10);   /* reset external status interrupts */
        break;

    case EXT_STAT_INT_A:

        outb(CH_A_CTL, 0x10);
        break;


    default:
        ;
    }
    EOI_80130_8274;
    EOI_8274;
}

Isr2()
{
    send_flag = TRUE;
    outb(0xEA, 125);
    outb(0xEA, 0x00); /* Timer 1 interrupts every .125 sec */
    outb(0xE0, 0x62);   /* EOI 80130 */
}
```

231421-57

/PCO/USR/CHUCK/CSRC/UAP.C

```
Load_Lsap()
{
    int     Recv_Data_1();

    for(;;) {
        Read_Addr("\n\nEnter this Station's LSAP in Hex --> ", &ssap, 1);
        if (!Add_Dsap_Address(ssap, Recv_Data_1)) {
            Write("\n\nError: LSAP Address must be one of the following:\n");
            Write("\n       20H, 40H, 60H, 80H, A0H, C0H, E0H \n");
        }
        else break;
    }
}

Load_Multicast()
{

    for ( ; ; ) {
        Read_Addr("\nEnter the Multicast Address in Hex -->",
                                            &Multi_Addr[0], ADD_LEN);
        if ((Multi_Addr[0] & 0x01) == 0)
            Write("\nSorry, the LSB of the Multicast Address must be 1\n");
        else { if (!Add_Multicast_Address(&Multi_Addr[0])) {
                    Write("\n\nSorry, Multicast Address Table is full!\n");
                    break;
                }
                else {
                    Write("\n\nWould you like to add another Multicast Address?");
                    Write(" (Y or N) --> ");
                    if (!Yes())
                        break;
                }
        }
    }
}

Remove_Multicast()
{

    for ( ; ; ) {
        Read_Addr("\nEnter the Multicast Address that you want to delete in Hex -->",
                                            &Multi_Addr[0], ADD_LEN);
        if ((Multi_Addr[0] & 0x01) == 0)
            Write("\nSorry, the LSB of the Multicast Address must be 1\n");
        else { if (!Delete_Multicast_Address(&Multi_Addr[0])) {
                    Write("\n\nSorry, that Multicast Address doesn't exist!\n");
                    break;
                }
                else {
                    Write("\n\nWould you like to delete another Multicast Address?");
                    Write(" (Y or N) --> ");
                    if (!Yes())
                        break;
                }
        }
    }
}
```

231421-58

```
    if (stat)   {
        Write("\n The following Multicast Addresses are enabled: ");
        for (pmat = &mat[0]; pmat <= &mat[MULTI_ADDR_CNT - 1]; pmat++)
            if (pmat->stat == INUSE) {
                Write_Addr(&pmat->addr[0], ADD_LEN);
                Write("                                                    ");
            }
    }
    else
        Write("\n There are no Multicast Addresses enabled.\n");
}

Init_DataLink()
{
    int     stat;

    if ((stat = Init_Llc()) == PASSED)
        Write("\n\nPassed Diagnostic Self Tests\n\n\n");
    else
        if(stat == FAILED_DIAGNOSE)
            Write("\n\nFailed: Self Test Diagnose Command\n");
    else
        if(stat == FAILED_LPBK_INTERNAL)
            Write("\n\nFailed: Internal Loopback Self Test\n");
    else
        if(stat == FAILED_LPBK_EXTERNAL)
            Write("\n\nFailed: External Loopback Self Test\n");
    else
        if(stat == FAILED_LPBK_TRANSCEIVER)
            Write("\n\nFailed: External Loopback Through Transceiver Self Test\n");
}

Init_Uap()
{

    outb(0xE0, 0x31);    /*initalize 80130 pic - ICW1 */
    outb(0xE2, 0x20);    /* ICW2 */
```

231421-59

```
/PCD/USR/CHUCK/CSRC/UAP.C

    outb(0xE2, 0x10);    /* ICW3 */
    outb(0xE2, 0x0D);    /* ICW4 */
    outb(0xE2, 0x10);    /* ICW6 */
    outb(0xE2, 0xFF);    /* mask all interrupts */

    outw(0xFF20, 0x0020);    /* set 80186 vector base */

    /* Initialize the 80130 timers for Terminal Mode */

    outb(0xEE, 0x34);
    outb(0xE8, 0xB8);
    outb(0xE8, 0x0B); /* SYSTICK set for 1 msec */
    outb(0xEE, 0x70);
    outb(0xEA, 125);
    outb(0xEA, 0x00); /* Timer 1 interrupts every .125 sec */

    /* Initialize the 8274 */
    outb(CH_B_CTL, 0x10); outb(CH_B_CTL, 0x28); outb(CH_B_CTL, 0x30);
    outb(CH_A_CTL, 0x38);
    outb(CH_B_CTL, 2); outb(CH_B_CTL, wrb[2] = 0x14);
    outb(CH_B_CTL, 1); outb(CH_B_CTL, wrb[1] = 0x15);
    outb(CH_B_CTL, 5); outb(CH_B_CTL, wrb[5] = 0xEA);

    Write("\n\n\n\n\n\n\n\n\n\n\n\n\n");
    Write("          ************************************************\n");
    Write("          * 82586 IEEE 802.2/802.3 Compatible Data Link Driver *\n");
    Write("          ************************************************\n");
    Write("\n\n\n\n\n\n\n\n");

    Init_DataLink();

    dhex = FALSE;
    monitor_flag = TRUE;

    Read_Addr("\n\nEnter the Address of the Destination Node in Hex --> ",
                                        &Dest_Addr[0], ADD_LEN);

    Load_Lsap();

    Read_Addr("\n\nEnter the Destination Node's LSAP in Hex --> ", &dsap, 1);

    Write("\n\nDo you want to Load any Multicast Addresses? (Y or N) -->");

    if (Yes())
        Load_Multicast();

    Print_Addresses();
}

Terminal_Mode()
{
    int     frame_cnt, buf_cnt;
    struct  TBD     *ptbd, *q, *begin_ptbd;
    char            *pbuf, c;

    Write("\n Would you like the local echo on? (Y or N)-->");

    if(Yes())
```

231421-60

/PCD/USR/CHUCK/CSRC/UAP.C

```
        local_echo = TRUE;
    else
        local_echo = FALSE;

    Write("\n This program will now enter the terminal mode.\n\n");
    Write("\n Press ^C then CR to return back to the menu\n\n\n");

    /* Initialize Fifo variables */

    out_fifo_t = in_fifo_t = out_fifo_r = in_fifo_r = 0;
    t_buf_stat = EMPTY;  r_buf_stat = EMPTY;

    EOI_80130_8274;
    Enable_Uart_Int();
    Enable_Timer_Int();
    monitor_flag = FALSE;
    tmstat = TRUE;
    while (tmstat)  {

        for (frame_cnt = 0; frame_cnt < MAX_FRAME_SIZE;  q = ptbd) {

            while ((ptbd = Get_Tbd()) == pNULL); /* get a xmit buffer from the
                                                    data link */
            pbuf = (char *) ptbd->buff_ptr;  /* point to the buffer */
            buf_cnt = 0;
            if (frame_cnt == 0) { /* if this is the first buffer, add on IEEE 802.2
                                     header information */
                begin_ptbd = ptbd;
                *pbuf++ = dsap;
                *pbuf++ = ssap;
                *pbuf++ = UI;
                buf_cnt = 3;
            }
            else  q->link = Offset(ptbd); /* if this isn't the first buffer
                                             link the previous buffer with the new one */
            /* fill up a data link xmit buffer from async transmit fifo */
            for ( ; buf_cnt < TBUF_SIZE && frame_cnt < MAX_FRAME_SIZE;
                                        buf_cnt++, pbuf++, frame_cnt++) {
                if (frame_cnt != 0 && send_flag)
                    break;

                while (t_buf_stat == EMPTY);     /* wait until fifo has data */
                if ((c = *pbuf = Fifo_T_Out()) == CR) {
                    ++buf_cnt ; ++pbuf; ++frame_cnt;
                    break;
                }
            }
            if (c == CR || buf_cnt < TBUF_SIZE || send_flag) { /* last buffer in list */
                ptbd->act_cnt = buf_cnt | EOFBIT;
                send_flag = FALSE;
                break;
            }
        }
        while(!Send_Frame(begin_ptbd, &Dest_Addr[0])); /* keep trying until
                                                          successful */
    }
```

231421-61

```
/PCO/USR/CHUCK/CSRC/UAP.C

    Disable_Uart_Int();
    Disable_Timer_Int();
    monitor_flag = TRUE;
}

struct  TBD     *Build_Frame(cnt)
    u_short     cnt;
{
    u_short         buf_cnt, frame_cnt, i;
    struct  TBD     *ptbd, *q, *begin_ptbd;
    char            *pbuf;

    i = 0x20; frame_cnt = 0;

    for ( ; ; q = ptbd) {

        while ((ptbd = Get_Tbd()) == pNULL); /* get a xmit buffer from the
                                                data link */
        pbuf = (char *) ptbd->buff_ptr;  /* point to the buffer */
        buf_cnt = 0;

        if (frame_cnt == 0) { /* if this is the first buffer, add on IEEE 802.2
                                 header information */
            begin_ptbd = ptbd;
            *pbuf++ = dsap;
            *pbuf++ = ssap;
            *pbuf++ = UI;
            buf_cnt = 3;
        }
        else  q->link = Offset(ptbd); /* if this isn't the first buffer
                                 link the previous buffer with the new one */
        /* fill up a data link xmit buffer with ASCII characters */
        for (; buf_cnt < TBUF_SIZE && cnt > 0;
                                 i++, buf_cnt++, pbuf++, cnt--, frame_cnt++) {
            *pbuf = i;
            if (i > 0x7E)
                i = 0x1F;
        }
        if (cnt == 0) { /* last buffer in list */
            ptbd->act_cnt = buf_cnt | EOFBIT;
            break;
        }
    }
    return(begin_ptbd);
}

Monitor_Mode()
{
    u_short     xmit, cnt, i;
    struct  TBD     *Build_Frame(), *ptbd;

    Write(" Do you want this station to transmit? (Y or N) --> ");
    if (Yes())
```

231421-62

```
/PCO/USR/CHUCK/CSRC/UAP.C

        for (xmit = FALSE; xmit == FALSE; ) {
        Write("\n Enter the number of data bytes in the frame --> ");
        cnt = Read_Int();
        if (cnt > 2045)
            Write ("\n Sorry, the number has to be less than 2046!\n");
        else
            xmit = TRUE;
        }

    else xmit = FALSE;

    Write("\n Hit any key to exit Monitor Mode.\n\n");

Write("  # of Good      # of Good      CRC       Alignment      No          Receive\n");
Write("   Frames          Frames      Errors     Errors         Resource    Overrun\n");
Write(" Transmitted      Received                              Errors      Errors\n");

   /* "01234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
         xxxxxxxxxx     xxxxxxxxxx     xxxx       xxxx         xxxx          xxxx
           xxxxxxxx       xxxxxxxx
              11             25        33          44           57           71 */

    for (i = 0; i < 79; i++)
        line[i] = 0x20;
    line[79] = CR;
    line[80] = '\0';

    while ((inb(CH_B_CTL) & 1) == 0) {
        for (i = 0; i < 72; i++)
            line[i] = 0x20;
        Write_Long_Int(good_xmit_cnt, 11);
        Write_Long_Int(recv_frame_cnt, 25);
        Write_Short_Int(scb.crc_errs, 33);
        Write_Short_Int(scb.aln_errs, 44);
        Write_Short_Int(scb.rsc_errs, 57);
        Write_Short_Int(scb.ovr_errs, 71);
        Write(&line[0]);
        if (xmit) {
            ptbd = Build_Frame(cnt);
            while(!Send_Frame(ptbd, &Dest_Addr[0]));
        }
    }
    i = Ci();
}

Hs_Xmit_Mode()
{
    struct  TBD  *ptbd;

    Write("\n Hit any key to exit High Speed Transmit Mode.\n\n");

    hs_stat = TRUE;
    EOI_80130_8274;
    Enable_Uart_Int();

    /* Execute this loop until a recv char interrupt happends at Uart */
```

231421-63

```
/PCO/USR/CHUCK/CSRC/UAP.C

    while (hs_stat) {
        while ((ptbd = Get_Tbd()) == pNULL); /* get a xmit buffer from
                                                the data link */
        ptbd->act_cnt |= EOFBIT;          /* set the End Of Frame bit */
        while(!Send_Frame(ptbd, &Dest_Addr[0])); /* Send Frame */
    }

    Disable_Uart_Int();
}

Print_Cnt()
{
    char    ch[11], base, dwidth, width, i;
    unsigned    long    temp;

        (dhex) {
        dwidth = 8;
        width = 4;
        base = 16;
    }
    else {
        base = 10;
        dwidth = 10;
        width = 5;
    }

    Write("\n\n Good frames transmitted: ");
    for (i = 1; i <= 11 - dwidth; i++)
        Co(SP);
    Int_To_Ascii(good_xmit_cnt, base, ' ', &ch[0], dwidth);
    for (i = dwidth - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  Good frames received: ");
    for (i = 1; i <= 15 - dwidth; i++)
        Co(SP);
    Int_To_Ascii(recv_frame_cnt, base, ' ', &ch[0], dwidth);
    for (i = dwidth - 1; i >= 0; i--)
        Co(ch[i]);
    Write("\n\n CRC errors received: ");
    for (i = 1; i <= 15 - width; i++)
        Co(SP);
    temp = scb.crc_errs;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  Alignment errors received: ");
    for (i = 1; i <= 10 - width; i++)
        Co(SP);
    temp = scb.aln_errs;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("\n\n Out of Resource frames: ");
    for (i = 1; i <= 12 - width; i++)
        Co(SP);
    temp = scb.rsc_errs;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
```

231421-64

/PCO/USR/CHUCK/CBRC/UAP.C

```
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  Receiver overrun frames: ");
    for (i = 1; i <= 12 - width; i++)
        Co(SP);
    temp = scb.ovr_errs;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("\n\n 82586 Reset: ");
    for (i = 1; i <= 23 - width; i++)
        Co(SP);
    temp = reset_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  Transmit underrun frames: ");
    for (i = 1; i <= 11 - width; i++)
        Co(SP);
    temp = underrun_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("\n\n Lost CRS: ");
    for (i = 1; i <= 26 - width; i++)
        Co(SP);
    temp = no_crs_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  SQE errors: ");
    for (i = 1; i <= 25 - width; i++)
        Co(SP);
    temp = sqe_err_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("\n\n Maximum retry: ");
    for (i = 1; i <= 21 - width; i++)
        Co(SP);
    temp = max_col_cnt;
    Int_To_Ascii(temp, base, ' ', &ch[0], width);
    for (i = width - 1; i >= 0; i--)
        Co(ch[i]);
    Write("  Frames that deferred: ");
    for (i = 1; i <= 15 - dwidth; i++)
        Co(SP);
    Int_To_Ascii(defer_cnt, base, ' ', &ch[0], dwidth);
    for (i = dwidth - 1; i >= 0; i--)
        Co(ch[i]);
}


Print_Help()
{
    Write ("\n\n Commands are:\n\n");
    Write (" T - Terminal Mode                    M - Monitor Mode\n");
```

231421-65

/PCO/USR/CHUCK/CSRC/UAP.C

```
    Write (" X - High Speed Transmit Mode          V - Change Transmit Statistics\n");
    Write (" P - Print All Counters                C - Clear All Counters\n");
    Write (" A - Add a Multicast Address           Z - Delete a Multicast Address\n");
    Write (" S - Change the SSAP Address           D - Change the DSAP Address\n");
    Write (" N - Change Destination Node Address   L - Print All Addresses\n");
    Write (" R - Re-Initialize the Data Link       B - Change the number Base\n");
}

Main()
{
    int     c;

    Init_Uap();
    Print_Help();

    for (;;) {
        Write ("\n\n Enter a command, type H for Help --> ");
        c = Read_Char();
        switch (Lower_Case(c)) {

        case 'h':
            Print_Help();
            break;
        case 'm':
            Monitor_Mode();
            break;
        case 't':
            Terminal_Mode();
            break;
        case 'x':
            Hs_Xmit_Mode();
            break;
        case 'v':
            Write("\n Transmit Statistics are now ");
            if (flags.stat_on == 1)
                Write("on.\n Would you like to change it ? (Y or N) --> ");
            else
                Write("off.\n Would you like to change it ? (Y or N) --> ");
            if (Yes()) {
                if (flags.stat_on == 1)
                    flags.stat_on = 0;
                else flags.stat_on = 1;
            }
            break;
        case 'p':
            Print_Cnt();
            break;
        case 'c':
            Clear_Cnt();
            break;
        case 'a':
            Load_Multicast();
            break;
        case 'z':
            Remove_Multicast();
            break;
        case 's':
```

231421-66

/PCO/USR/CHUCK/CSRC/UAP.C

```
            Delete_Dsap_Address(ssap);
            Load_Lsap();
            break;
        case 'd':
            Read_Addr("\n\nEnter the Destination Node's LSAP in Hex --> ", &dsap, 1);
            break;
        case 'n':
            Read_Addr("\n\nEnter the Address of the Destination Node in Hex --> ",
                                                    &Dest_Addr[0], ADD_LEN);
            break;
        case 'l':
            Print_Addresses();
            break;
        case 'r':
            Software_Reset();
            Init_DataLink();
            Add_Dsap_Address(ssap, Recv_Data_1);
            break;
        case 'b':
            Write("\n The current base is ");
            if (dhex == TRUE)
                Write("Hex.\n Would you like to change it ? (Y or N) --> ");
            else
                Write("Decimal.\n Would you like to change it ? (Y or N) --> ");
            if (Yes()) {
                if (dhex == TRUE)
                    dhex = FALSE;
                else dhex = TRUE;
            }
            break;

        default:
            Write ("\n Unknown command\n");
            break;
        }
    }
}
```

231421-67

```
/PCO/USR/CHUCK/CSRC/ASSY.ASM

name    c_assy_support

stack   segment stack   'stack'
stktop  label   word
stack   ends

DLD_DATA                segment public  'DATA'
extrn   SEGMT_:word                     ; data segment address
DLD_DATA                ends

UAP_DATA        segment public   'DATA'
UAP_DATA        ends

DLD_CODE        segment public  'CODE'
        extrn    Isr_Timeout_:far, Isr_586_:far, Isr7_:far
        extrn    Isr6_:far, Isr5_:far, Isr1_:far
DLD_CODE        ends

UAP_CODE        segment public  'CODE'
        extrn   Isr_Uart_:far, Isr2_:far, Main_:far
UAP_CODE        ends


DG_CODE         segment public  'CODE'

        public   inw_, outw_, init_intv_, enable_, disable_, Build_Ptr_
        public   Offset_, begin, inb_, outb_

arg1    equ     [BP + 6]
arg2    equ     [BP + 8]

        assume  CS:DG_CODE
        assume  DS:DLD_DATA

;+
;       initialization program for the 82586 data link driver
;-

begin:

        sti
        mov     ax, DLD_DATA ;get base of dgroup and
        mov     SEGMT_, ax      ;pass the segment value to the c program
        mov     ds, ax
        call Main_              ;go to the c program
        hlt


inb_    proc    far
        push    BP
        mov     BP, SP
        push    DX
        mov     DX, arg1
        in      AL, DX
        pop     DX
        mov     SP, BP
```

231421-68

/PCO/USR/CHUCK/CSRC/ASSY.ASM

```
        pop     BP
        ret
inb_    endp

outb_   proc    far
        push    BP
        mov     BP, SP
        push    DX
        push    AX
        mov     DX, arg1
        mov     AX, arg2
        out     DX, AL
        pop     AX
        pop     DX
        mov     SP, BP
        pop     BP
        ret
outb_   endp

inw_    proc    far
        push    BP
        mov     BP, SP
        push    DX
        mov     DX, arg1
        in      AX, DX
        pop     DX
        mov     SP, BP
        pop     BP
        ret
inw_    endp

outw_   proc    far
        push    BP
        mov     BP, SP
        push    DX
        push    AX
        mov     DX, arg1
        mov     AX, arg2
        out     DX, AX
        pop     AX
        pop     DX
        mov     SP, BP
        pop     BP
        ret
outw_   endp

Build_Ptr_      proc    far
        push    BP
        mov     BP, SP
        mov     DX, DLD_DATA
        mov     AX, arg1
        mov     SP, BP
        pop     BP
        ret
Build_Ptr_      endp

Offset_ proc    far
```

231421-69

```
        mov     AX, DGD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr_386_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int_isr   endp

serve_int_8274  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, UAP_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr_Uart_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
```

231421-70

```
/PCD/USR/CHUCK/CSRC/ASSY.ASM

        pop     CX
        pop     BX
        pop     AX
        iret
serve_int_8274  endp

serve_int_timeout       proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, DLD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr_Timeout_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int_timeout       endp

serve_int7_isr  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, DLD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr7_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
```

231421-71

```
/PCO/USR/CHUCK/CSRC/ASSY.ASM


        iret
serve_int7_isr  endp

serve_int6_isr  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, DLD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr6_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int6_isr  endp

serve_int5_isr  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, DLD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr5_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int5_isr  endp
```

231421-72

```
/PCO/USR/CHUCK/CSRC/ASSY.ASM

serve_int2_isr  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, UAP_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr2_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int2_isr  endp

serve_int1_isr  proc    far
        push    AX
        push    BX
        push    CX
        push    DX
        push    SI
        push    DI
        push    DS
        push    ES

        mov     AX, DLD_DATA
        mov     DS, AX
        mov     ES, AX

        call    Isr1_

        pop     ES
        pop     DS
        pop     DI
        pop     SI
        pop     DX
        pop     CX
        pop     BX
        pop     AX
        iret
serve_int1_isr  endp

enable_ proc    far
        sti
```

231421-73

```
        mov     DS:word ptr  82h, DG_CODE
        mov     DS:word ptr  84h, offset serve_int1_isr         ; int 1
        mov     DS:word ptr  86h, DG_CODE
        mov     DS:word ptr  88h, offset serve_int2_isr         ; int 2
        mov     DS:word ptr  8Ah, DG_CODE
        mov     DS:word ptr  8Ch, offset serve_int_isr          ; int 3
        mov     DS:word ptr  8Eh, DG_CODE
        mov     DS:word ptr  90h, offset serve_int_timeout      ; int 4
        mov     DS:word ptr  92h, DG_CODE
        mov     DS:word ptr  94h, offset serve_int5_isr         ; int 5
        mov     DS:word ptr  96h, DG_CODE
        mov     DS:word ptr  98h, offset serve_int6_isr         ; int 6
        mov     DS:word ptr  9Ah, DG_CODE
        mov     DS:word ptr  9Ch, offset serve_int7_isr         ; int 7
        mov     DS:word ptr  9Eh, DG_CODE


        pop     AX
        pop     DS
        ret

init_intv_      endp

DG_CODE ends

        end             begin, ds:dld_data, ss:stack:stktop
```

231421-74

intel®

APPLICATION NOTE

AP-236

November 1986

Implementing StarLAN with the Intel 82588

ADI GOLBERT
DATA COMMUNICATIONS OPERATION

SHARAD GANDHI
FIELD APPLICATIONS-EUROPE

Order Number: 231422-003

IMPLEMENTING StarLAN WITH THE INTEL 82588

CONTENTS

PAGE

1.0 INTRODUCTION 1-421
- 1.1 StarLAN 1-421
- 1.2 The 82588 1-421
- 1.3 Organization of the Application Note 1-421
- 1.4 References 1-421

2.0 StarLAN 1-421
- 2.1 StarLAN Topology 1-422
  - 2.1.1 Telephone Network 1-422
  - 2.1.2 StarLAN and the Telephone Network 1-423
  - 2.1.3 StarLAN and Ethernet 1-425
- 2.2 Basic StarLAN Components 1-425
  - 2.2.1 A StarLAN Node Interface 1-426
  - 2.2.2 StarLAN HUB 1-426
  - 2.2.3 StarLAN Cable 1-428
- 2.3 Framing 1-428
- 2.4 Signal Propagation and Collision 1-429
  - 2.4.1 Situation #1 1-431
  - 2.4.2 Situation #2 1-431
  - 2.4.3 Situation #3 1-431
- 2.5 StarLAN System And Network Parameters 1-431

3.0 LAN CONTROLLER FOR StarLAN 1-431
- 3.1 IEEE 802.3 Compatibility 1-431
- 3.2 Configurability of the 82588 1-431
- 3.3 Clocks and Timers 1-433
- 3.4 Manchester Data Encoding and Decoding 1-433
- 3.5 Detection of the Collision Presence Signal 1-434
  - 3.5.1 Collision Detection by Code Violation 1-435
  - 3.5.2 Collision Detection by Signature (or Bit) Comparison 1-435
  - 3.5.3 Additional Collision Detection Mechanisms 1-436

CONTENTS

PAGE

3.6 Carrier Sensing 1-436
3.7 Squelching the Input 1-437
3.8 System Bus Interface 1-438
3.9 Debug and Diagnostic Aids 1-439
3.10 Jitter Performance 1-440

4.0 THE 82588 1-440
4.1 Transmit and Retransmit Operations 1-441
4.2 Configuring the 82588 1-441
4.3 Frame Reception 1-441
4.3.1 Multiple Buffer Frame Reception 1-442
4.4 Memory Dump of Registers 1-444
4.5 Other Operations 1-444

5.0 StarLAN NODE FOR IBM PC 1-444
5.1 Interfacing to the IBM PC I/O Channel 1-447
5.1.1 Register Access and Data Bus Interface 1-447
5.1.2 Control Port 1-448
5.1.3 Clock Generation 1-449
5.1.4 DMA Interface 1-449
5.1.5 Interrupt Controller 1-450
5.2 Serial Link Interface 1-451
5.2.1 Transmit Path 1-451
5.2.2 Idle Pattern Generation 1-451
5.3 Receive Path 1-452
5.4 80188 Interface to 82588 1-454
5.5 iSBX Interface to StarLAN 1-454

6.0 THE StarLAN HUB 1-457
6.1 A StarLAN HUB for the IBM/PC 1-457

CONTENTS

PAGE

6.1.1 HUB Input Ports 1-465
6.1.2 Collision Detection 1-466
6.1.3 The Local 82588 1-467
6.1.4 The Collision Presence Signal 1-467
6.1.5 Signal Retiming 1-468
6.1.6 Retiming Circuit, Theory of Operation 1-468
6.1.7 Retiming Circuit Implementation 1-469
6.1.8 Driver Circuits 1-471
6.1.9 Header/Intermediate HUB Switch 1-471
6.1.10 Jabber Function 1-471
6.1.11 HUB Receiver Protection Timer 1-472
6.1.12 HUB Reliability 1-472

7.0 SOFTWARE DRIVER 1-473
7.1 Interfacing to IBM PC 1-473
7.1.1 Doing I/O on IBM PC 1-473
7.2 Initialization and Declarations 1-475
7.3 General Commands 1-477
7.4 DMA Routines 1-480
7.5 Interrupt Routines 1-482

APPENDIX A: StarLAN SIGNALS 1-485

APPENDIX B: 802.3 1BASE5 MULTI-POINT EXTENSION (MPE) 1-488

APPENDIX C: SINGLE DMA CHANNEL INTERFACE 1-490

APPENDIX D: MEASURING NETWORK DELAYS WITH THE 82588 1-492

1.0 INTRODUCTION

Personal computers have become the most prolific workstation in the office, serving a wide range of needs such as word processing, spreadsheets, and data bases. The need to interconnect PCs in a local environment has clearly emerged, for purposes such as the sharing of file, print, and communication servers; downline loading of files and application programs; electronic mail; etc. Proliferation of the PC makes it the workstation of choice for accessing the corporate mainframe/s; this function can be performed much more efficiently and economically when clusters of PCs are already interconnected through Local Area Networks (LANs). According to market surveys, the installed base of PCs in business environments reached about 10 million units year-end '85, with only a small fraction connected via LANs. The installed base is expected to double by 1990. There is clearly a great need for locally interconnecting these machines; furthermore, end users expect interconnectability across vendors. Thus, there is an urgent need for industry standards to promote cost effective PC LANs.

A large number of proprietary PC LANs have become available for the office environment over the past several years. Many of these suffer from high installed cost, technical deficiencies, non-conformance to industry standards, and general lack of industry backing. StarLAN, in Intel's opinion, is one of the few networks which will emerge as a standard. It utilizes a proven network access method, it is implemented with proven VLSI components; it is cost effective, easily installable and reconfigurable; it is technically competent; and it enjoys the backing of a large cross section of the industry which is collaborating to develop a standard (IEEE 802.3, type 1BASE5).

1.1 StarLAN

StarLAN is a 1 Mb/s network based on the CSMA/CD access method (Carrier Sense, Multiple Access with Collision Detection). It works over standard, unshielded, twisted pair telephone wiring. Typically, the wiring connects each desk to a wiring closet in a star topology (from which the IEEE Task Force working on the standard derived the name StarLAN in 1984). In fact, telephone and StarLAN wiring can coexist in the same twisted pair bundle connecting a desk to the wiring closet. Abundant quantities of unused phone wiring exist in most office environments, particularly in the U.S. The StarLAN concept of wiring and networking concepts was originated by AT&T Information Systems.

1.2 The 82588

The 82588 is a single-chip LAN controller designed for CSMA/CD networks. It integrates in one chip all functions needed for such networks. Besides inplementing the standard CSMA/CD functions like framing, deferring, backing off and retrying on collisions, transmitting and receiving frames, it performs data encoding and decoding in Manshester or NRZI format, carrier sensing and collision detection, all up to a speed of 2 Mb/s (independent of the chosen encoding scheme). These functions make it an optimum controller for a StarLAN node. The 82588 has a very conventional microcomputer bus interface, easing the job of interfacing it to any processor.

1.3 Organization of the Application Note

This application note has two objectives. One is to describe StarLAN in practical terms to prospective implementers. The other is to illustrate designing with 82588, particularly as related to StarLAN which is expected to emerge as its largest application area.

Section 2 of this Application Note describes the StarLAN network, its basic components, collision detection, signal propagation and network parameters. Sections 3 and 4 describe the 82588 LAN controller and its role in the StarLAN network. Section 5 goes into the details of designing a StarLAN node for the IBM PC. Section 6 describes the design of the HUB. Both these designs have been implemented and operated in an actual StarLAN environment. Section 7 documents the software used to drive the 82588. It gives the actual procedures used to do operations like, configure, transmit and receive frames. It also shows how to use the DMA controller and interrupt controller in the IBM PC and goes into the details of doing I/O on the PC using DOS calls. Appendix A shows oscilloscope traces of the signals at various points in the network. Appendix B describes the multiple point extension (MPE) being considered by IEEE. Appendixes C and D talk about advanced usages of the 82588; working with only one DMA channel, and measuring network delays with the 82588.

1.4 References

For additional information on the 82588, see the Intel Microcommunications Handbook. StarLAN specification are currently available in draft standard form through the IEEE 802.3 Working Group.

2.0 StarLAN

StarLAN is a low cost 1 Mb/s networking solution aimed at office automation applications. It uses a star

topology with the nodes connected in a point-to-point fashion to a central HUB. HUBs can be connected in a hierarchical fashion. Up to 5 levels are supported. The maximum distance between a node and the adjacent HUB or between two adjacent HUBs is 800 ft. (about 250 meters) for 24 gauge wire and 600 ft. (about 200 meters) for 26 gauge wire. Maximum node-to-node distance with one HUB is 0.5 km, hence IEEE 802.3 designation of type 1BASE5. 1 stands for 1 Mb/s and BASE for baseband. (StarLAN doesn't preclude the use of more than 800 ft wiring provided 6.5 dB maximum attenuation is met, and cable propagation delay is no more than 4 bit times).

One of the most attractive features of StarLAN is that it uses telephone grade twisted pair wire for the transmission medium. In fact, existing installed telephone wiring can also be used for StarLAN. Telephone wiring is very economical to buy and install. Although use of telephone wiring is an obvious advantage, for small clusters of nodes, it is possible to work around the use of building wiring.

Factors contributing to low cost are:

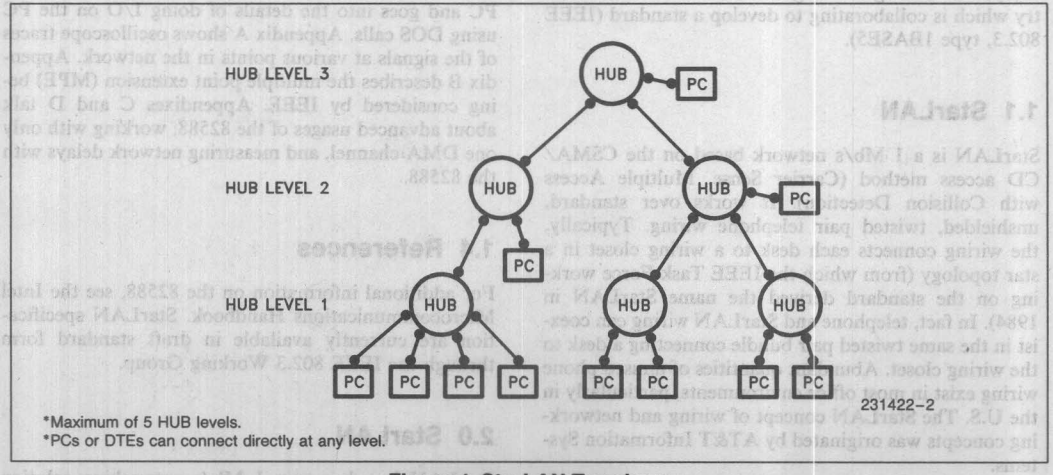
1) Use of telephone grade, unshielded, 24 or 26 gauge twisted pair wire transmission media.
2) Installed base of redundant telephone wiring in most buildings.
3) Buildings are designed for star topology wiring. They have conduits leading to a central location.
4) Availability of low cost VLSI LAN controllers like the 82588 for low cost applications and the 82586 for high performance applications.
5) Off-the-shelf, Low cost RS-422, RS-485 drivers/receivers compatible with the StarLAN analog interface requirements.

2.1 StarLAN Topology

StarLAN, as the name suggests, uses a star topology. The nodes are at the extremities of a star and the central point is called a HUB. There can be more than one HUB in a network. The HUBs are connected in a hierarchical fashion resembling an inverted tree, as shown in Figure 1, where nodes are shown as PCs. The HUB at the base (at level 3) of the tree is called the Header Hub (HHUB) and others are called Intermediate HUBs (IHUB). It will become apparent, later in this section, that topologically, this entire network of nodes and HUBs is equivalent to one where all the nodes are connected to a single HUB. Also StarLAN doesn't limit the number of nodes or HUBS at any given level.

2.1.1 TELEPHONE NETWORK

StarLAN is structured to run parallel to the telephone network in a building. The telephone network has, in fact, exactly the same star topology as StarLAN. Let us now examine how the telephone system is typically laid out in a building in the USA. Figure 2 shows how a typical building is wired for telephones. 24 gauge unshielded twisted pair wires emanate from a Wiring Closet. The wires are in bundles of 25 or 50 pairs. The bundle is called D inside wiring (DIW). The wires in these cables end up at modular telephone jacks in the wall. The telephone set is either connected directly to



*Maximum of 5 HUB levels.
*PCs or DTEs can connect directly at any level.

Figure 1. StarLAN Topology

the jack or through an extension cable. Each telephone generally needs one twisted pair for voice and another for auxilliary power. Thus, each modular jack has 2 twisted pairs (4 wires) connected to it. A 25 pair DIW cable can thus be used for up to 12 telephone connections. In most buildings, not all pairs in the bundle are used. Typically, a cable is used for only 4 to 8 telephone connections. This practice is followed by telephone companies because it is cheaper to install extra wires initially, rather than retrofitting to expand the existing number of connections. As a result, a lot of extra, unused wiring exists in a building. The stretch of cable between the wiring closet and the telephone jack is typically less than 800 ft. (250 meters). In the wiring closet the incoming wires from the telephones are routed to another wiring closet, a PABX or to the central office through an interconnect matrix. Thus, the wiring closet is a concentration point in the telephone network. There is also a redundancy of wires between the wiring closets.

2.1.2 StarLAN AND THE TELEPHONE NETWORK

StarLAN does not have to run on building wiring, but the fact that it can significantly adds to its attractiveness. Figure 3 shows how StarLAN piggybacks on telephone wiring. Each node needs two twisted pair wires to connect to the HUB. The unused wires in the 25 pair DIW cables provide an electrical path to the wiring closet, where the HUB is located. Note that the telephone and StarLAN are electrically isolated. They only use the wires in the same bundle cable to connect to the wiring closet. Within the wiring closet, StarLAN wires connect to a HUB and telephone wires are routed to a different path. Similar cable sharing can occur in connecting HUBs to one another. See Figure 4 for a typical office wired for StarLAN through telephone wiring.

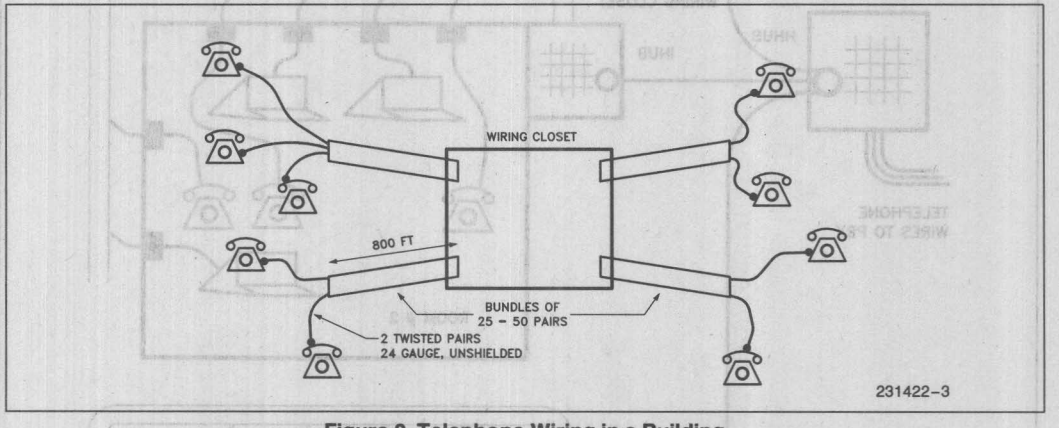
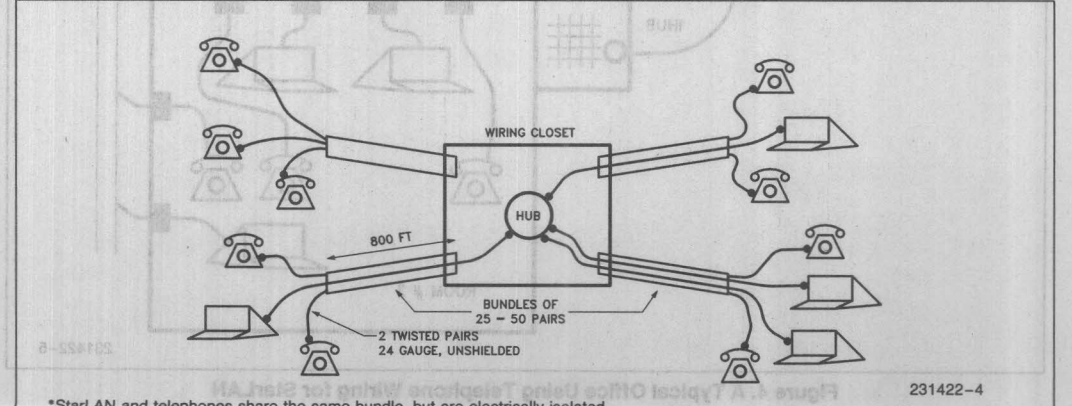


Figure 2. Telephone Wiring in a Building



*StarLAN and telephones share the same bundle, but are electrically isolated.
*StarLAN uses the unused wires in existing bundles.

Figure 3. Coexistence of Telephone and StarLAN

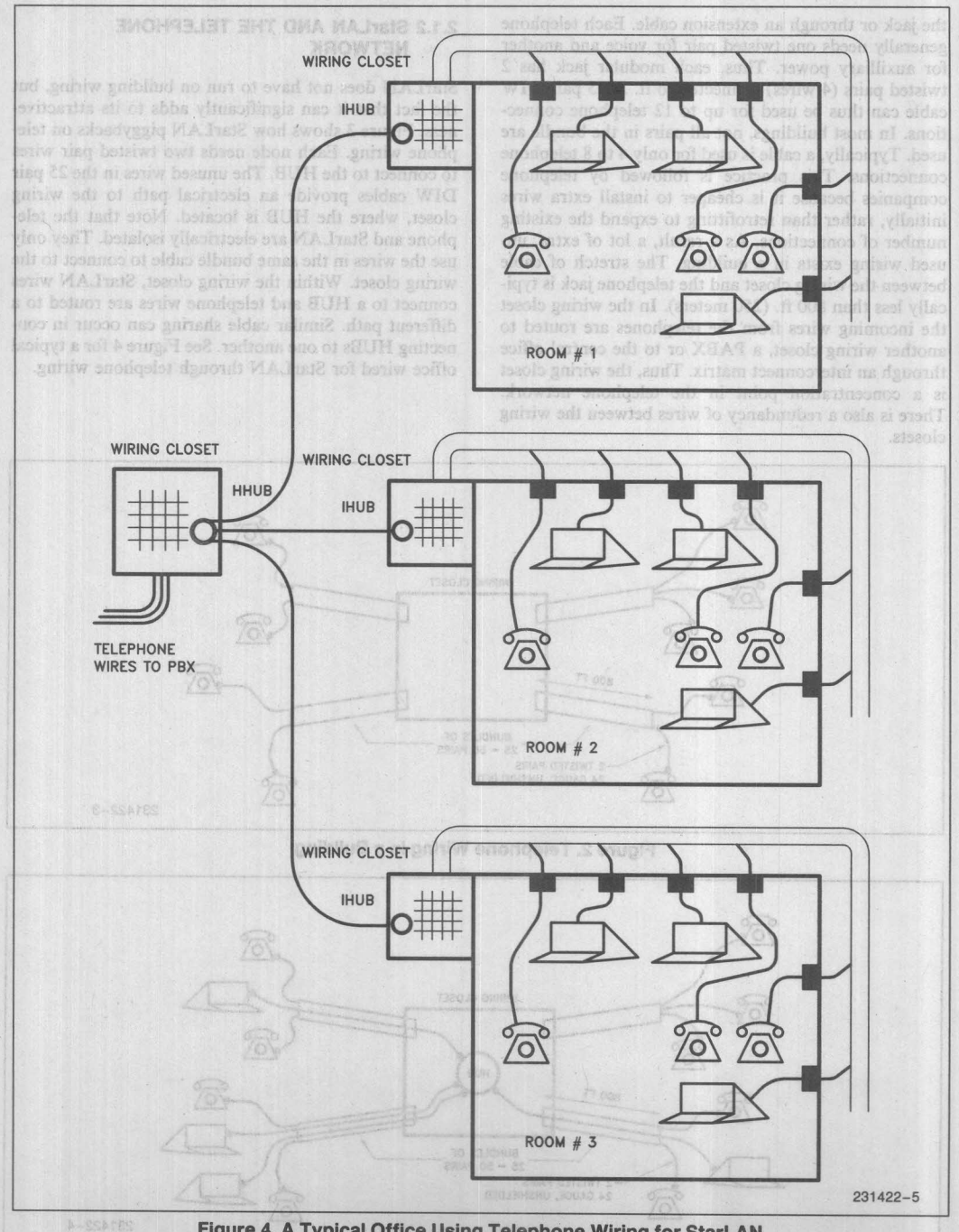


Figure 4. A Typical Office Using Telephone Wiring for StarLAN

2.1.3 StarLAN AND Ethernet

StarLAN and Ethernet are similar CSMA/CD networks. Since Ethernet has existed longer and is better understood, a comparison of Ethernet with StarLAN is worthwhile.

1. The data rate of Ethernet is 10Mb/s and that of StarLAN is 1 Mb/s.
2. Ethernet uses a bus topology with each node connected to a coaxial cable bus via a 50 meter transceiver cable containing four shielded twisted pair wires. StarLAN uses a star topology, with each node connected to a central HUB by a point to point link through two pairs of unshielded twisted pair wires.
3. Collision detection in Ethernet is done by the transceiver connected to the coaxial cable. Electrically, it is done by sensing the energy level on the coax cable. Collision detection in StarLAN is done in the HUB by sensing activity on more than one input line connected to the HUB.
4. In Ethernet, the presence of collision is signalled by the transceiver to the node by a special collision detect signal. In StarLAN, it is signalled by the HUB using a special collision presence signal on the receive data line to the node.
5. Ethernet cable segments are interconnected using repeaters in a non-hierarchical fashion so that the distance between any two nodes does not exceed 2.8 kilometers. In StarLAN, the maximum distance between any two nodes is 2.5 kilometers. This is achieved by wiring a maximum of five levels of HUBs in a hierarchical fashion.

2.2 Basic StarLAN Components

A StarLAN network has three basic components:

1. StarLAN node interface
2. StarLAN HUB
3. Cable

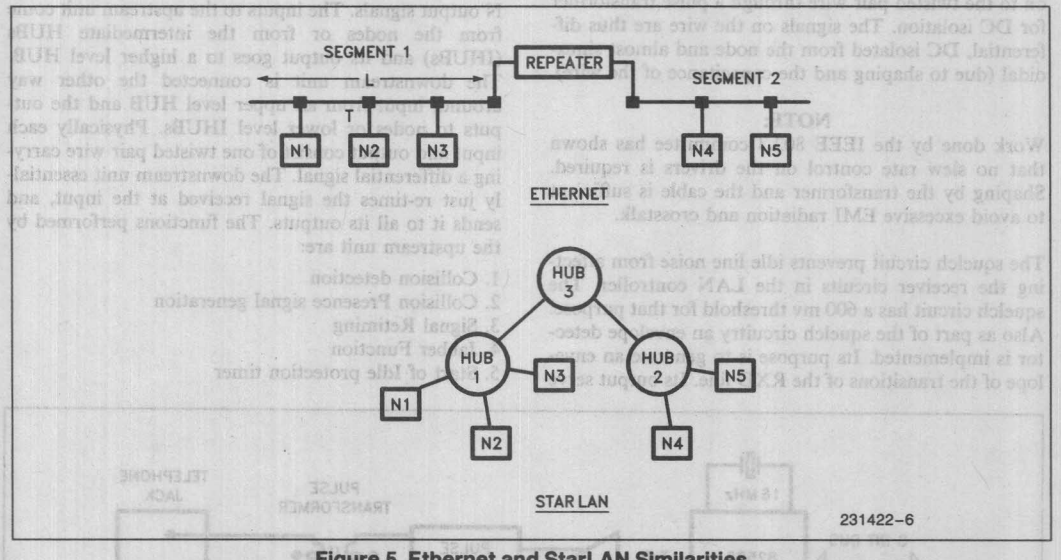


Figure 5. Ethernet and StarLAN Similarities

2.2.1 A StarLAN NODE INTERFACE

Figure 6 shows a typical StarLAN node interface. It interfaces to a processor on the system side. The processor runs the networking software. The heart of the node interface is the LAN controller which does the job of receiving and transmitting the frames in adherence to the IEEE 802.3 standard protocol. It maintains all the timings—like the slot time, interframe spacing etc.—required by the network. It performs the functions of framing, deferring, backing-off, collision detection which are necessary in a CSMA/CD network. It also does Manchester encoding of data to be transmitted and clock separation—or decoding—of the Manchester encoded data that is received. These signals before going to the unshielded twist pair wire, may undergo pulse shaping (optional) pulse shaping basically slows down the fall/rise times of the signal. The purpose of that is to diminish the effects of cross-talk and radiation on adjacent pairs sharing the same bundle (digital voice, T1 trunks, etc). The shaped signal is sent on to the twisted pair wire through a pulse transformer for DC isolation. The signals on the wire are thus differential, DC isolated from the node and almost sinusoidal (due to shaping and the capacitance of the wire).

NOTE:
Work done by the IEEE 802.3 committee has shown that no slew rate control on the drivers is required. Shaping by the transformer and the cable is sufficient to avoid excessive EMI radiation and crosstalk.

The squelch circuit prevents idle line noise from affecting the receiver circuits in the LAN controller. The squelch circuit has a 600 mv threshold for that purpose. Also as part of the squelch circuitry an envelope detector is implemented. Its purpose is to generate an envelope of the transitions of the RXD line. Its output serve as a carrier sense signal. The differential signal from the HUB is received using a zero-crossing RS-422 receiver. Output of the receiver, qualified by the squelch circuit, is fed to the RxD pin of the LAN controller. The RxD signal provides three kinds of information:

1) Normal received data, when receiving the frame.
2) Collision information in the form of the collision presence signal from the HUB.
3) Carrier sense information, indicating the beginning and the end of frame. This is useful during transmit and receive operations.

2.2.2 StarLAN HUB

HUB is the point of concentration in StarLAN. All the nodes transmit to the HUB and receive from the HUB. Figure 7 shows an abstract representation of the HUB. It has an upstream and a downstream signal processing unit. The upstream unit has N signal inputs and 1 signal output. And the downstream unit has 1 input and N output signals. The inputs to the upstream unit come from the nodes or from the intermediate HUBs (IHUBs) and its output goes to a higher level HUB. The downstream unit is connected the other way around; input from an upper level HUB and the outputs to nodes or lower level IHUBs. Physically each input and output consist of one twisted pair wire carrying a differential signal. The downstream unit essentially just re-times the signal received at the input, and sends it to all its outputs. The functions performed by the upstream unit are:

1. Collision detection
2. Collision Presence signal generation
3. Signal Retiming
4. Jabber Function
5. Start of Idle protection timer

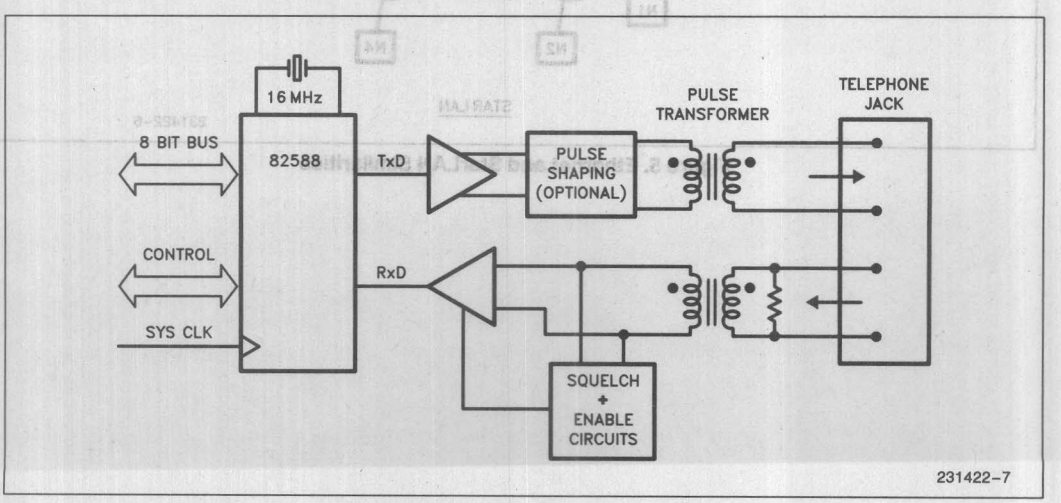


Figure 6. 82588 Based StarLAN Node

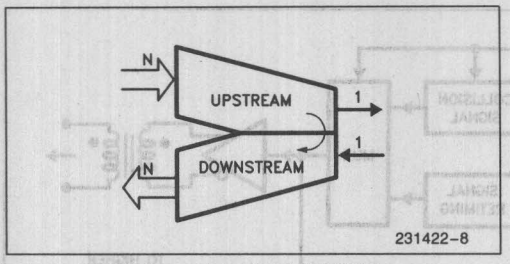


Figure 7. A StarLAN HUB

The collision detection in the HUB is done by sensing the activity on the inputs. If there is activity (or transitions) on more than one input, it is assumed that more than one node is transmitting. This is a collision. If a collision is detected, a special signal called the Collision Presence Signal is generated. This signal is generated and sent out as long as activity is sensed on any of the input lines. This signal is interpreted by every node as an occurrence of collision. If there is activity only on one input, that signal is re-timed—or cleaned up of any accumulated jitter—and sent out. Figure 8 shows the input to output relations of the HUB as a black box.

If a node transmits for too long the HUB exercises a Jabber function to disable the node from interfering with traffic from other nodes. There are two timers in the HUB associated with this function and their operation is described in section 6.

The last function implemented by the HUB is the start of Idle protection timer. During the end of reception, the HUB will see a long undershoot at its input port. This undershoot is a consequence of the transformer discharging accumulated charge during the 2 microseconds of high of the idle pattern. The HUB should implement a protection mechanism to avoid the undesirable effects of that undershoot.

Figure 9 shows a block diagram of the HUB. A switch position determines whether the HUB is an IHUB or a HHUB (Header HUB). If the HUB is an IHUB, the switch decouples the upstream and the downstream units. HHUB is the highest level HUB; it has no place to send its output signal, so it returns its output signal (through the switch) to the outputs of the downstream unit. There is one and only one HHUB in a StarLAN network and it is always at the base of the tree. The returned signal eventually reaches every node in the network through the intermediate nodes (if any). StarLAN specifications do not put any restrictions on the number of IHUBS at any level or on number of inputs to any HUB. The number of inputs per HUB are typically 6 to 12 and is dictated by the typical size of clusters in a given networking environment.

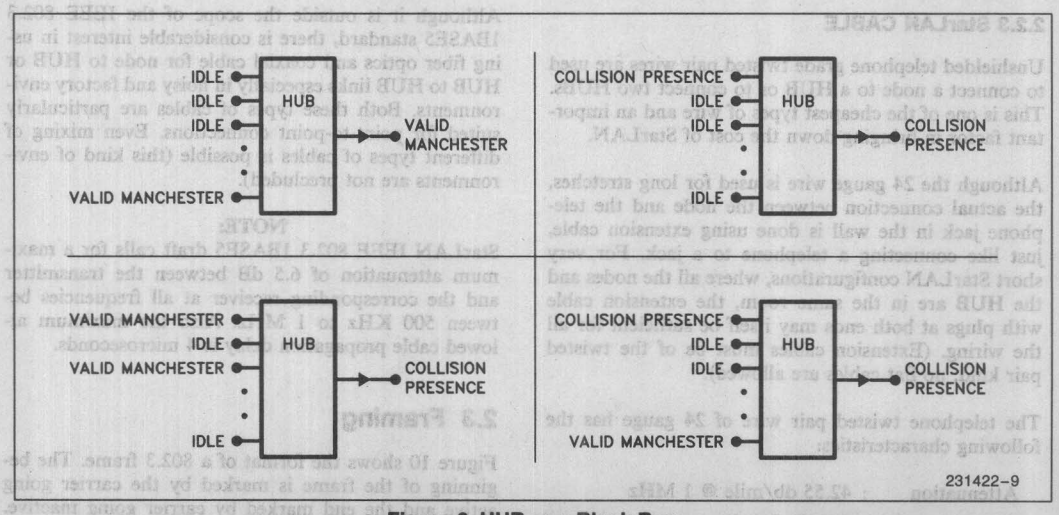


Figure 8. HUB as a Black Box

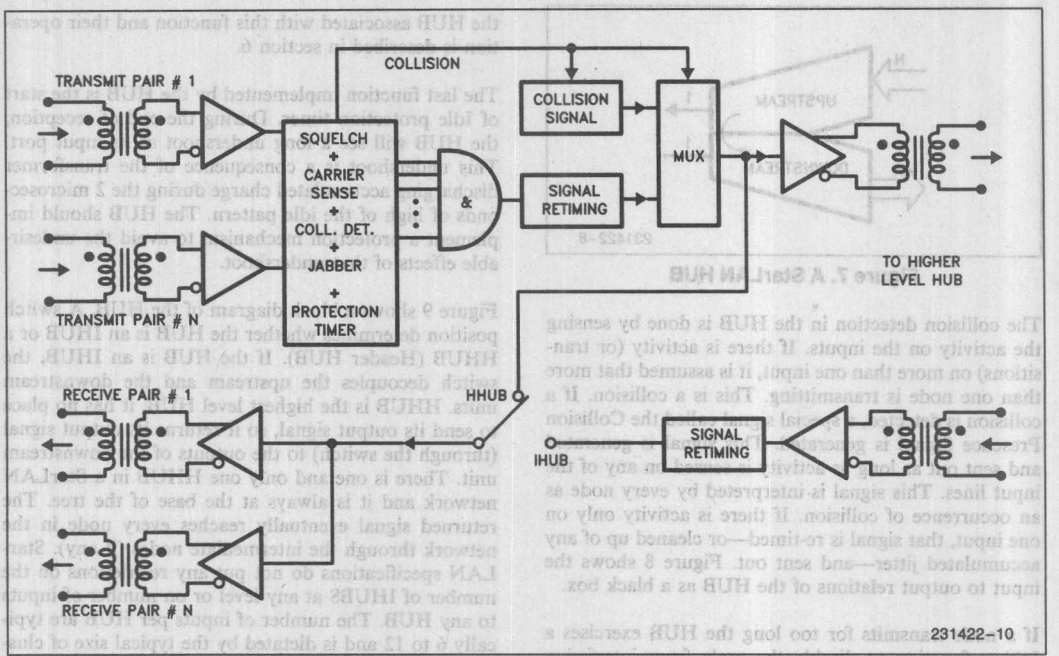


Figure 9. StarLAN HUB Block Diagram

2.2.3 StarLAN CABLE

Unshielded telephone grade twisted pair wires are used to connect a node to a HUB or to connect two HUBs. This is one of the cheapest types of wire and an important factor in bringing down the cost of StarLAN.

Although the 24 gauge wire is used for long stretches, the actual connection between the node and the telephone jack in the wall is done using extension cable, just like connecting a telephone to a jack. For very short StarLAN configurations, where all the nodes and the HUB are in the same room, the extension cable with plugs at both ends may itself be sufficient for all the wiring. (Extension cables must be of the twisted pair kind, no flat cables are allowed).

The telephone twisted pair wire of 24 gauge has the following characteristics:

Attenuation	: 42.55 db/mile @ 1 MHz
DC Resistance	: 823.69 Ω/mile
Inductance	: 0.84 mH/mile
Capacitance	: 0.1 μF/mile
Impedance	: 92.6Ω, −4 degrees @ 1 MHz

Experiments have shown that the sharing of the telephone cable with other voice and data services does not cause any mutual harm due to cross-talk and radiation, provided every service meets the FCC limits.

Although it is outside the scope of the IEEE 802.3 1BASE5 standard, there is considerable interest in using fiber optics and coaxial cable for node to HUB or HUB to HUB links especially in noisy and factory environments. Both these types of cables are particularly suited for point-to-point connections. Even mixing of different types of cables is possible (this kind of environments are not precluded).

NOTE:
StarLAN IEEE 802.3 1BASE5 draft calls for a maximum attenuation of 6.5 dB between the transmitter and the corresponding receiver at all frequencies between 500 KHz to 1 MHz. Also the maximum allowed cable propagation delay is 4 microseconds.

2.3 Framing

Figure 10 shows the format of a 802.3 frame. The beginning of the frame is marked by the carrier going active and the end marked by carrier going inactive. The preamble has a 56 bit sequence of 101010 . . . . ending in a 0. This is followed by 8 bits of start of frame delimiter (sfd) − 10101011. These bits are transmitted with the MSB (leftmost bit) transmitted first. Source and destination fields are 6 bytes long. The first byte is the least significant byte. These fields are transmitted with LSB first. The length field is 2 bytes long and gives the length of data in the Information field. The entire information field is a minimum of 46 bytes and a maximum of 1500 bytes. If the data content of the Informa-

tion field is less than 46, padding bytes are used to make the field 46 bytes long. The Length field indicates how much real data is in the Information field. The last 32 bits of the frame is the Frame Check Sequence (FCS) and contains the CRC for the frame. The CRC is calculated from the beginning of the destination address to the end of the Information field. The generating polynomial (Autodin II) used for CRC is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

No need for Figure N.

The frames can be directed to a specific node (LSB of address must be 0), to a group of nodes (multicast or group—LSB of address must be 1) or all nodes (broadcast—all address bits must be 1).

2.4 Signal Propagation and Collision

Figure 11 will be used to illustrate three typical situations in a StarLAN with two IHUBs and one HHUB. Nodes A and B are connected to HUB1, nodes C and D to HUB2 and node E to HUB3.

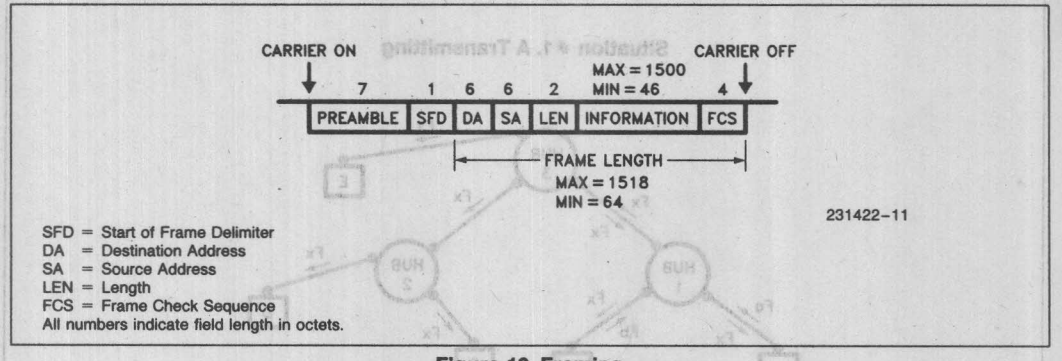
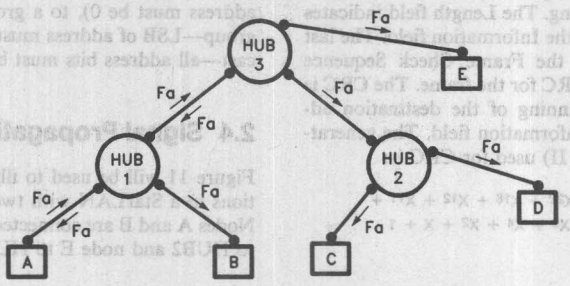
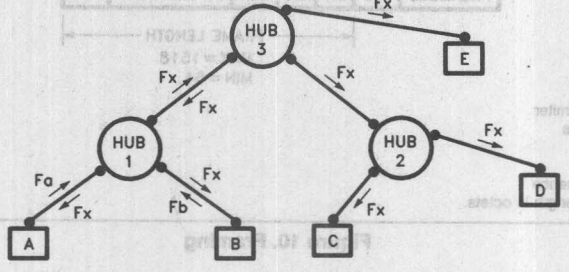


Figure 10. Framing



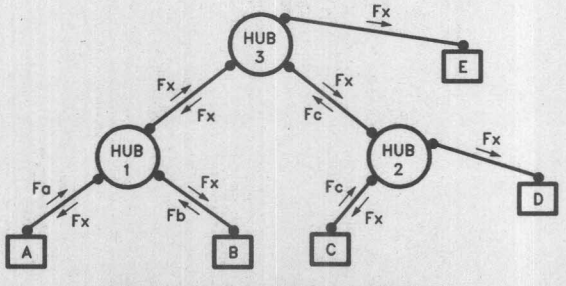
231422-12

Situation #1. A Transmitting



231422-13

Situation #2. A & B Transmitting



231422-14

Situation #3. A, B & C Transmitting

HUB1, HUB2 are IHUBs
HUB3 is the HHUB

Fa, Fb, Fc—Frames from nodes A, B & C
Fx—Collision Presence Signal

Figure 11. Signal Propagation and Collisions

2.4.1 Situation #1

Whenever node A transmits a frame Fa, it will reach HUB1. If node B is silent, there is no collision. HUB1 will send Fa to HUB3 after re-timing the signal. If nodes C, D and E are also silent, there is no collision at HUB2 or HUB3. Since HUB3 is the HHUB, it sends the frame Fa to HUB1, HUB2 and to node E after re-timing. HUB1 and HUB2 send the frame Fa to nodes A, B and C, D. Thus, Fa reaches all the nodes on the network including the originator node A. If the signal received by node A is a valid Manchester signal and not the Collision Presence Signal (CPS) for the entire duration of the slot time, then the node A assumes that it was a successful transmission.

2.4.2 Situation #2

If both nodes A and B were to transmit, HUB1 will detect it as a collision and will send signal Fx (the Collision Presence Signal) to the HUB3—Note that HUB1 does not send Fx to nodes A and B yet. HUB 3 receives a signal from HUB1 but nothing from node E or HUB2, thus it does not detect the situation as a collision and simply re-times the signal Fx and sends it to node E, HUB2 and HUB1. Fx ultimately reach all the nodes. Nodes A and B detect this signal as CPS and call it a collision.

2.4.3 Situation #3

In addition to nodes A and B, if node C were also to transmit, the situation at HUB1 will be the same as in situation #2. HUB2 will propagate Fc from C towards HUB3. HUB3 now sees two of its inputs active and hence generates its own Fx signal and sends it towards each node.

These situations should also illustrate the point made earlier in the chapter that, the StarLAN network, with nodes connected to multiple HUBs is, logically, equivalent to all the nodes connected to a single HUB (Yet there are some differences between stations connected at different HUB levels, those are due to different delays to the header hub HHUB).

2.5 StarLAN System and Network Parameters

Preamble length (incl. sfd)	64 bits
Address length	6 bytes
FCS length CRC (Autodin II)	32 bits
Maximum frame length	1518 bytes
Minimum frame length	64 bytes
Slot time	512 bit times
Interframe spacing	96 bit times
Minimum jam timing	32 bit times
Maximum number of collisions	16
Backoff limit	10
Backoff method	Truncated binary exponential
Encoding	Manchester
Clock tolerance	±0.01% (100 ppm)
Maximum jitter per segment	±62.5 ns

3.0 LAN CONTROLLER FOR StarLAN

One of the attractive features of StarLAN is the availability of the 82588, a VLSI LAN controller, designed to meet the needs of a StarLAN node. The main requirements of a StarLAN node controller are:

1. IEEE 802.3 compatible CSMA/CD controller.
2. Configurable to StarLAN network and system parameters.
3. Generation of all necessary clocks and timings.
4. Manchester data encoding and decoding.
5. Detection of the Collision Presence Signal.
6. Carrier Sensing.
7. Squelch or bad signal filtering.
8. Fast and easy interface to the processor.

82588 performs all these functions in silicon, providing a minimal hardware interface between the system processor and the StarLAN physical link. It also reduces the software needed to run the node, since a lot of functions, like deferring, back off, counting the number of collisions etc., are done in silicon.

3.1 IEEE 802.3 Compatibility

The CSMA/CD control unit on the 82588 performs the functions of deferring, maintaining the Interframe Space (IFS) timing, reacting to collision by generating a jam pattern, calculating the back-off time based on the number of collisions and a random number, decoding the address of the incoming frame, discarding a frame that is too short, etc. All these are performed by the 82588 in accordance to the IEEE 802.3 standards. For inter-operability of different nodes on the StarLAN network it is very important to have the controllers strictly adhere to the same standards.

3.2 Configurability of the 82588

Almost all the networking parameters are programmable over a wide range. This means that the StarLAN parameters form a subset of the total potential of the 82588. This is a major advantage for networks whose standards are being defined and are in a flux. It is also an advantage when carrying over the experience gained with the component in one network to other applications, with differing parameters (leveraging the design).

The 82588 is initialized or configured to its working environment by the CONFIGURE command. After the execution of this command, the 82588 knows its system and network parameters. A configure block in

memory is loaded into the 82588 by DMA. This block contains all the parameters to be programmed as shown in Figure 12. Following is a partial list of the parameters with the programmable range and the StarLAN value:

Parameter	Range	StarLAN Value
Preamble length	2, 4, 8, 16 bytes	8
Address length	0 to 6 bytes	6
CRC type	16, 32 bit	32
Minimum frame length	6 to 255 bytes	64
Interframe spacing	12 to 255 bit times	96
Slot time	1 to 2047 bit times	512
Number of retries	0 to 15	15
Data encoding	NRZI, Man., Diff. Man.	Manch.
Collision detection	Code viol., Bit comp.	Code Viol.

Beside these, there are many other options available, which may or may not apply to StarLAN:

Data sampling rate of 8 or 16
Operating in Promiscuous mode
Reception of Broadcast frames
Internal loopback operation
External loopback operation
Transmit without CRC
HDLC Framing
.
.
.

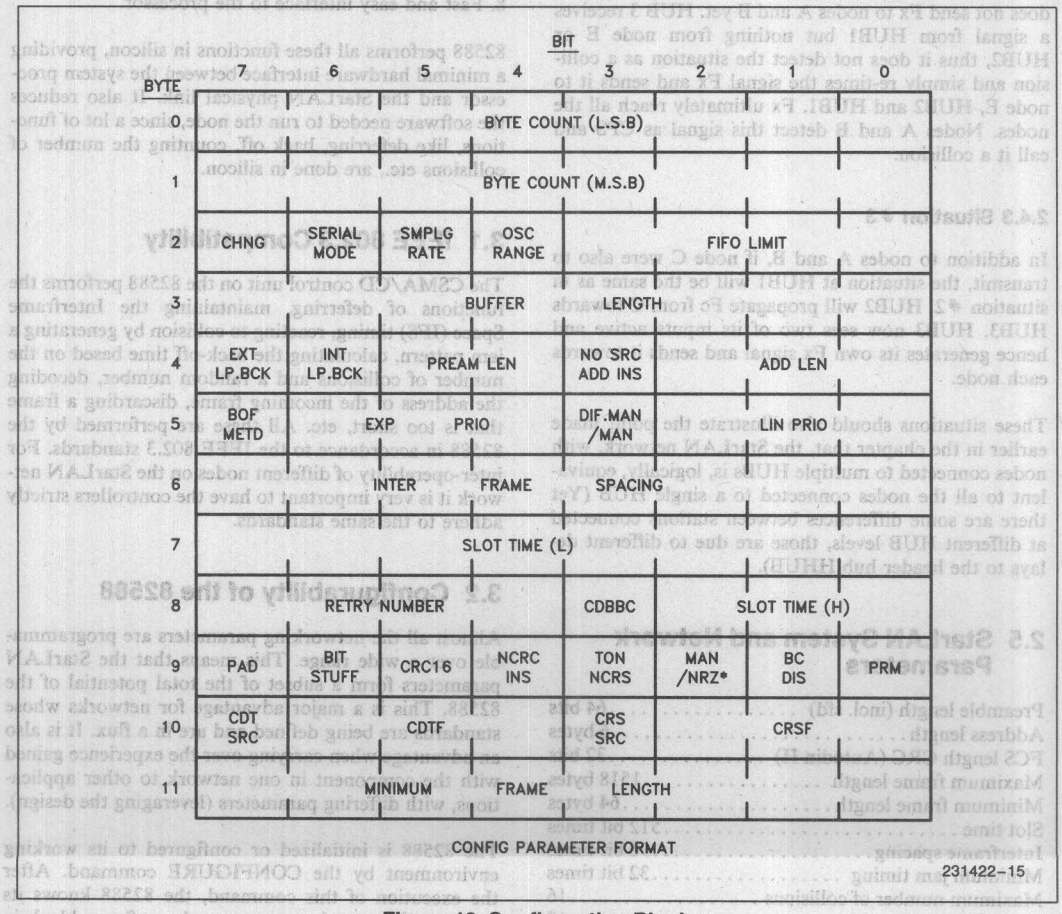


Figure 12. Configuration Block

3.3 Clocks and Timers

The 82588 requires two clocks; one for the operation of the system interface and another for the serial side. Both clocks are totally asynchronous to each other. This permits transmitting and receiving frames at data rates that are virtually independent of the speed at which the system interface operates.

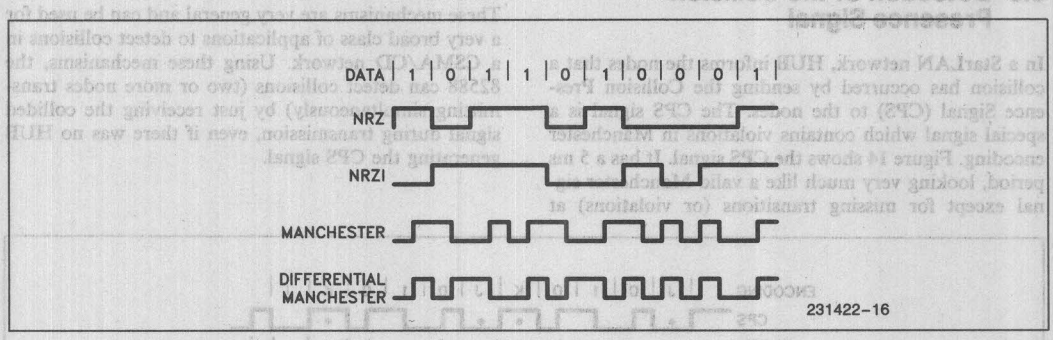
The serial clock can be generated on chip using just an external crystal of a value 8 or 16 times the desired bit rate. An external clock may also be used.

The 82588 has a set of timers to maintain various timings necessary to run the CSMA/CD control unit. These are timings for the Slot time, Interframe spacing time, Back off time, Number of collisions, Minimum frame length, etc. These timers are started and stopped automatically by the 82588.

3.4 Manchester Data Encoding and Decoding

In StarLAN the data transmitted by the node must be encoded in Manchester format. The node should also be able to decode Manchester encoded data when receiving a frame—a process also known as clock recovery. The 82588 does the encoding and decoding of data bits on chip for data rates up to 2 Mb/s.

Besides Manchester, the 82588 can also do encoding and decoding in NRZI and Differential Manchester formats. Figure 13 shows samples of encoding in



Encoding Method	Mid Bit Cell Transitions	Bit Cell Boundary Transitions
NRZ	Do not exist.	Identical to original data.
NRZI	Do not exist.	Exist only if original data bit equals 0. Dependent on present encoded signal level: to 0 if 1 to 1 if 0
Manchester	Exist for every bit of the original data: from 0 to 1 for 1 from 1 to 0 for 0	Exist for consequent equal bits of original data: from 1 to 0 for 1 1 from 0 to 1 for 0 0
Differential Manchester	Exist for every bit of the original data. Dependent on present Encoded signal level: to 0 if 1 to 1 if 0	Exist only if original data bit equals 0. Dependent on present Encoded signal level: to 0 if 1 to 1 if 0

Figure 13. 82588 Data Encoding Rules

these three formats. The main advantage of NRZI over the other two is that NRZI requires half the channel bandwidth, for any given data rate. On the other hand, since the NRZI signal does not have as many transitions as the other two, clock recovery from it is more difficult. The main advantage of Differential Manchester over straight Manchester is that for a signal that is differentially driven (as in RS 422), crossing of the two wires carrying the data does not change the data received at the receiver. In other words, NRZI and Differential Manchester encoding methods are polarity insensitive (Even though NRZI, Differential Manchester are polarity insensitive, the 82588 expects a high level in the RXD line to detect carrier inactive at the end of frames).

3.5 Detection of the Collision Presence Signal

In a StarLAN network, HUB informs the nodes that a collision has occurred by sending the Collision Presence Signal (CPS) to the nodes. The CPS signal is a special signal which contains violations in Manchester encoding. Figure 14 shows the CPS signal. It has a 5 ms period, looking very much like a valid Manchester signal except for missing transitions (or violations) at periodic intervals. When the 82588 decodes this signal, it fails to see mid-cell transitions repeatedly at intervals of 2.5 bit times and hence calls it a code violation. The edges of CPS are marked for illustration as a, b, c, d, . . . l. Let us see how the 82588 interprets the signal if it starts calling the edge 'a' as the mid-cell transition for '1'. Then edge at 'b' is '0'. Now the 82588 expects to see an edge at '*' but since there is none, it is a Manchester code violation. The edge that eventually does occur at 'd' is then used to re-synchronize and, since it is a falling edge, it is taken as a mid-cell transition for '0'. The edge at 'e' is for a '1' and then again there is no edge at '*'. This goes on, with the 82588 flagging code violation and re-synchronizing again every 2.5 bit times. When a transmitting node sees this CPS signal being returned by the HUB (instead of a valid Manchester signal it transmitted), it assumes that a collision occurred. The 82588 has two built-in mechanisms to detect collisions. These mechanisms are very general and can be used for a very broad class of applications to detect collisions in a CSMA/CD network. Using these mechanisms, the 82588 can detect collisions (two or more nodes transmitting simultaneously) by just receiving the collided signal during transmission, even if there was no HUB generating the CPS signal.

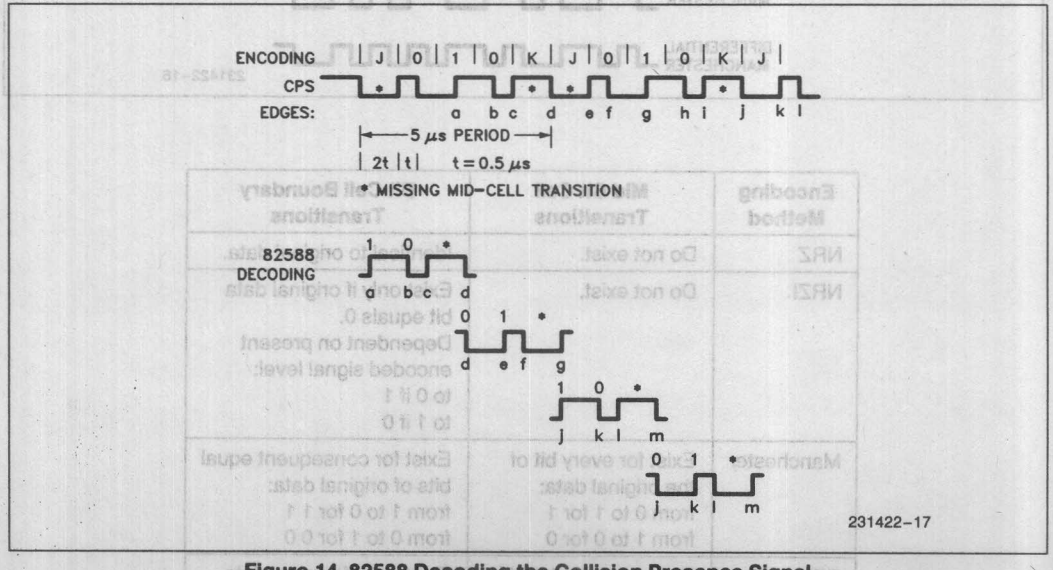
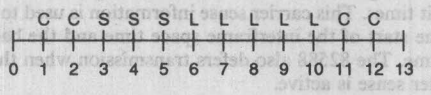


Figure 14. 82588 Decoding the Collision Presence Signal

3.5.1 COLLISION DETECTION BY CODE VIOLATION

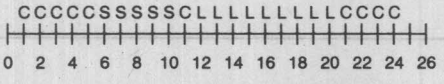
If during transmission, the 82588 sees a violation in the encoding (Manchester, NRZI or Differential Manchester) used, then it calls it a collision by aborting the transmission and transmitting a 32 bit jam pattern. The algorithm used to detect collisions, and to do the data decoding, is based on finding the number of sampling clocks between an edge to the next one. Suppose an edge occurred at time 0, the sampling instant of the next edge determines whether it was a collision (C), a long pulse (L)—with a nominal width of 1 bit time—,or a short pulse (S)—nominal width of half a bit time. The following two charts show the decoding and collision detection algorithm for sampling rates of 8 and 16 when using Manchester encoding. The numbers at the bottom of the line indicate sampling instances after the occurrence of the last edge (at 0). The alphabets on the top show what would be inferred by the 82588 if the next edge were to be there.

Sampling rate = 8 (clock is 8x bit rate)



Collision also if:
RxD stays low for 13 samples or more
A mid cell transition is missing

Sampling rate = 16 (clock is 16x bit rate)



Collision also if:
RxD stays low for 25 samples or more
A mid cell transition is missing

A single instance of code violation can qualify as collision. The 82588 has a parameter called collision detect filter (CDT Filter) that can be configured from 0 to 7. This parameter determines for how many bit times the violation must remain active to be flagged as a collision. For StarLAN CDT Filter must be configured to 0—that is disabled.

3.5.2 COLLISION DETECTION BY SIGNATURE (OR BIT) COMPARISON

This method of collision detection compares a signature of the transmitted data with that of the data received on the RxD pin while transmitting. Figure 15 shows a block diagram of the logic. As the frame is transmitted it flows through the CRC generation logic. A timer, called the Tx slot timer, is started at the same time that the CRC generation starts. When the count in the timer reaches the slot time value, the current value of the CRC generator is latched in as the transmit signature. As the frame is returned back (through the HUB) it flows through the CRC checker. Another timer—Rx slot timer—is started at the same time as the CRC checker starts checking. When this timer reaches the slot time value, the current value of the CRC checker is latched in as the receive signature. If the received signature matches the transmitted one, then it is assumed that there was no collision. Whereas, if the signatures do not match, a collision is assumed to have occurred.

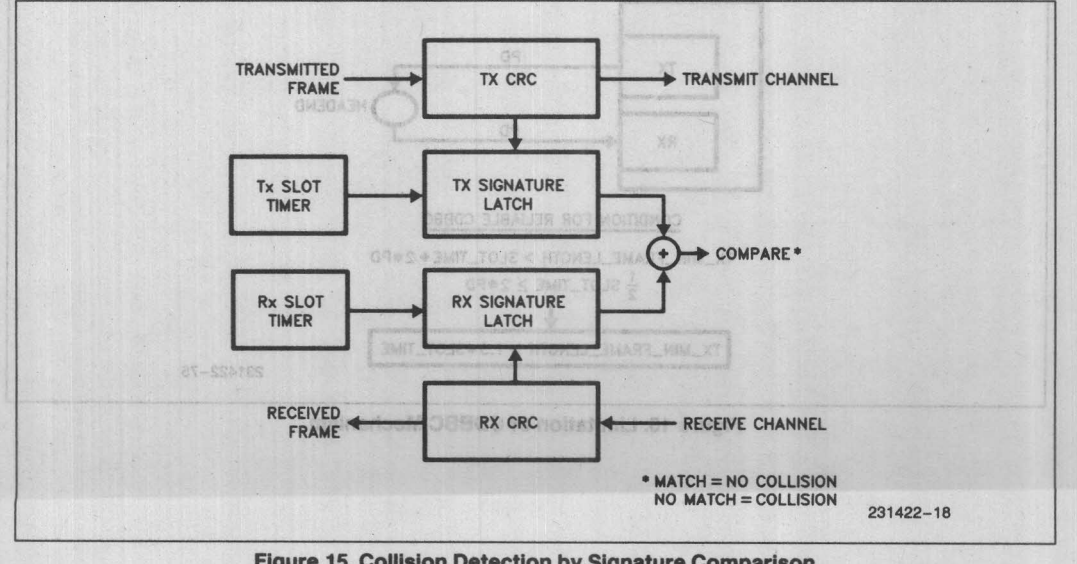


Figure 15. Collision Detection by Signature Comparison

has seen enough bits to complete its signature. Otherwise, the transmitter may be done before the RX signature is completed resulting in an undetected collision. A sufficient condition to avoid this situation is to transmit frames with a minimum length of 1.5 * slot-time (see Figure 16).

3.5.3 ADDITIONAL COLLISION DETECTION MECHANISM

In addition to the collision detection mechanisms described in the preceding sections, the 82588 also flags collision when after starting a transmission any of the following conditions become valid:

a) Half a slot time elapses and the carrier sense of 82588 is not active.

needs to know when the carrier is active and when not. This is done by the carrier sensing circuitry. On the 82588 this circuit is on chip. It looks at the RxD (receive data) pin and if there are transitions, it turns on an internal carrier sense signal. It turns off the carrier sense signal if RxD remains in idle (high) state for 13/8 bit times. This carrier sense information is used to mark the start of the interframe space time and the back off time. The 82588 also defers transmission when the carrier sense is active.

When operating in the NRZI encoded mode, carrier sense is turned off if RxD pin is in the idle state for 8 bit times or more (see Figure 18).

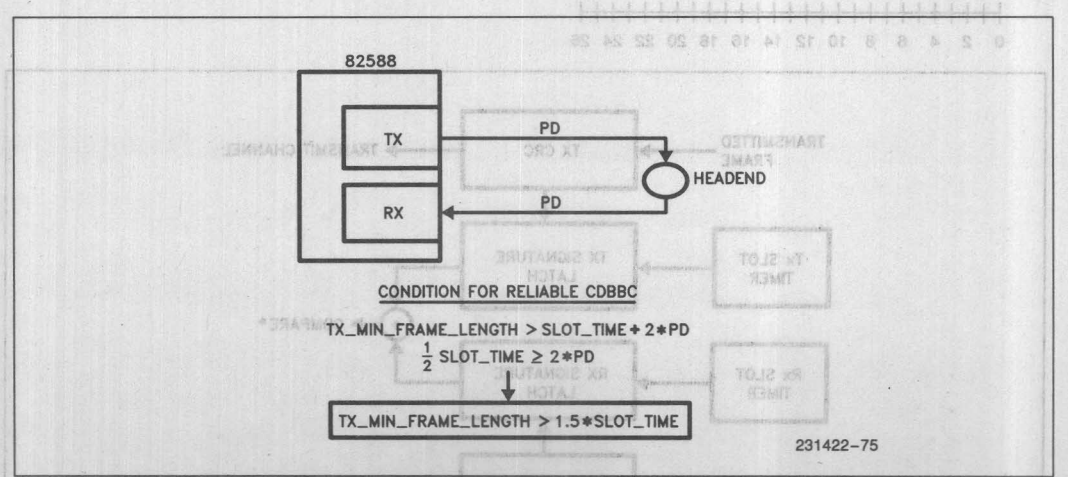


Figure 16. Limitation of CDBBC Mechanism

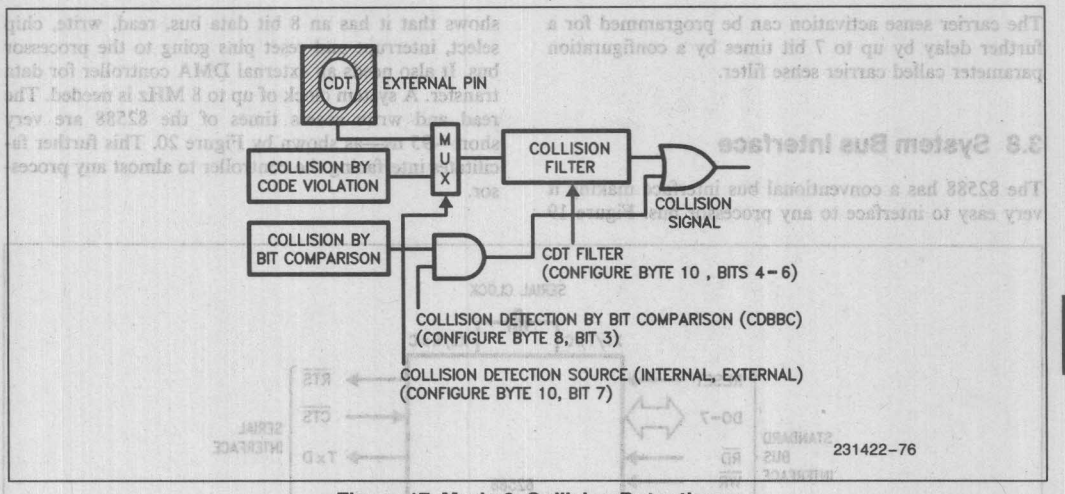


Figure 17. Mode 0, Collision Detection

3.7 Squelching the Input

Squelch circuit is used to filter idle noise on the receiver input. Basically two types of squelch may be used: Voltage and time. Voltage squelch is done to filter out signals whose strength is below a defined voltage threshold (0.6 volts for StarLAN). It prevents idle line noise from disturbing the receive circuits on the controller. The voltage squelch circuit is placed right after the receiving pulse transformer. It enables the input to the RxD pin of the 82588 only when the signal strength is above the threshold.

If the signal received has the proper level but not the proper timing, it should not bother the receiver. This is accomplished by the time squelch circuit on the 82588. Time squelching is essential to weed out spikes, glitches and bad signal especially at the beginning of a frame. The 82588 does not turn on its carrier sense (or receive enable) signal until it receives three consecutive edges, each separated by time periods greater than the fast time clock high time but less than 13/8 bit-times as shown in Figure 18.

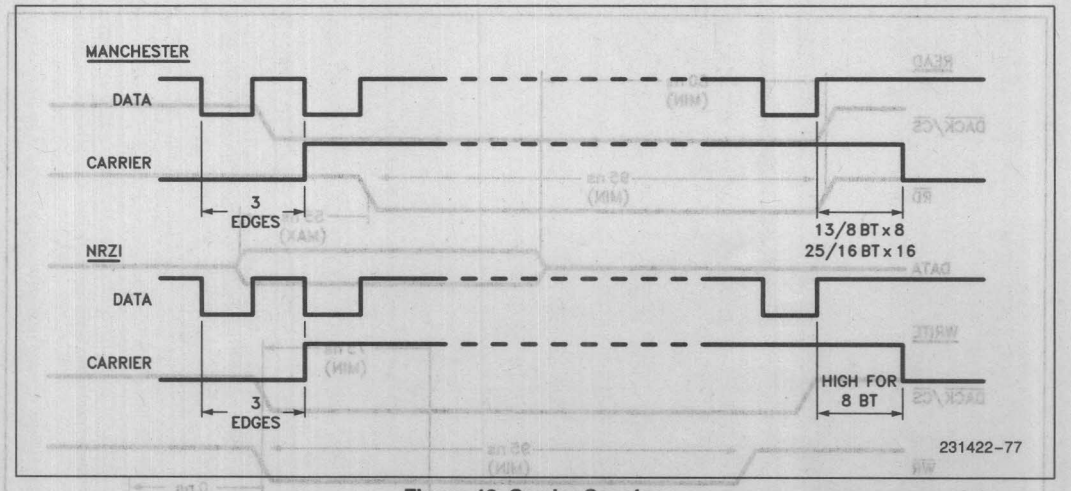


Figure 18. Carrier Sensing

The carrier sense activation can be programmed for a further delay by up to 7 bit times by a configuration parameter called carrier sense filter.

3.8 System Bus Interface

The 82588 has a conventional bus interface making it very easy to interface to any processor bus. Figure 19 shows that it has an 8 bit data bus, read, write, chip select, interrupt and reset pins going to the processor bus. It also needs an external DMA controller for data transfer. A system clock of up to 8 MHz is needed. The read and write access times of the 82588 are very short—95 ns—as shown by Figure 20. This further facilitates interfacing the controller to almost any processor.

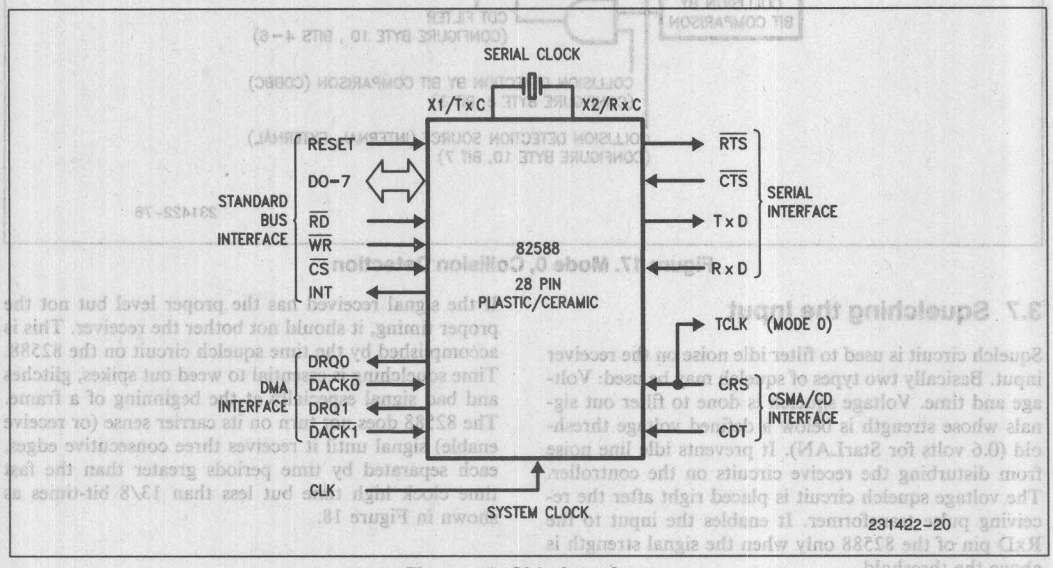


Figure 19. Chip Interface

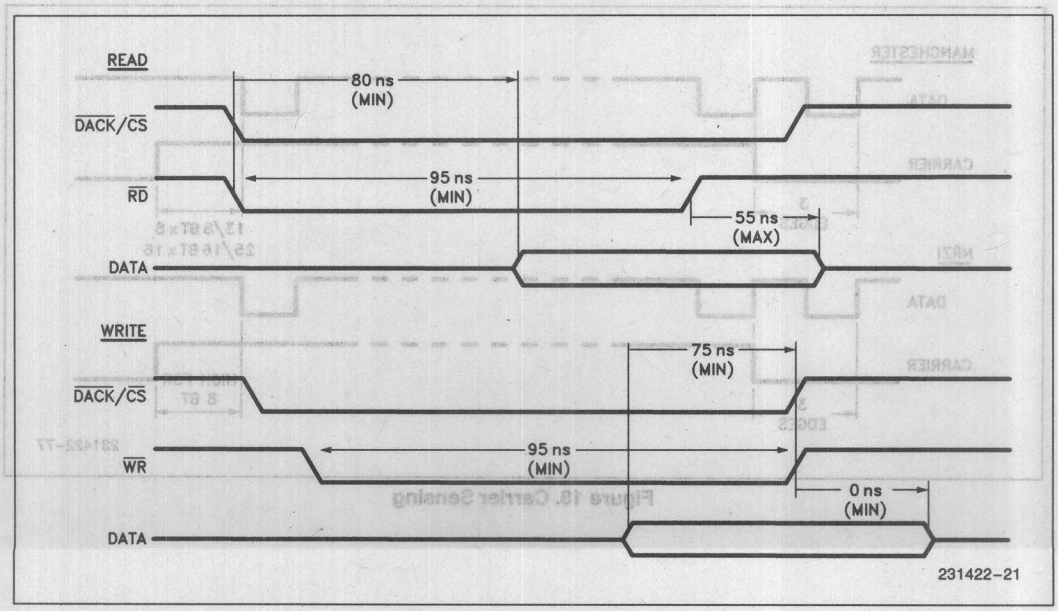


Figure 20. Access Times

The 82588 has over 50 bytes of registers, and most are accessed only indirectly. Figure 21 shows the register access mechanism of the 82588. It has one I/O port and 2 DMA channel ports. These are the windows into the 82588 for the CPU and the DMA controller. An external CPU can write into the Command register and read from the Status registers using I/O instructions and asserting chip select and write or read lines. Although there is just one I/O port and 4 status registers, they can be read out in a round robin fashion through the same port as shown in Figure 22. Other registers like the Configuration, Individual Address registers can be accessed only through DMA. All the internal registers can be dumped into memory by DMA using the Dump command. The execution of some of the commands is described in section 4. See the 82588 Reference Manual for details on these commands.

3.9 Debug and Diagnostic Aids

Besides the standard functions that can be used directly for StarLAN, the 82588 offers many debug and diag-

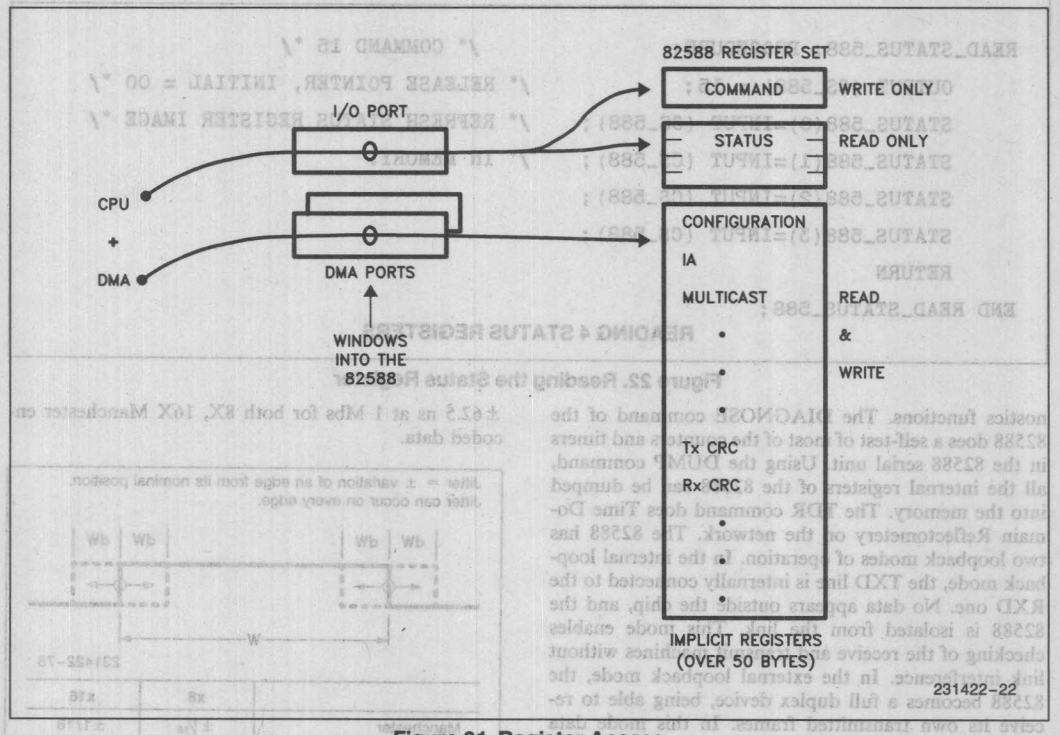
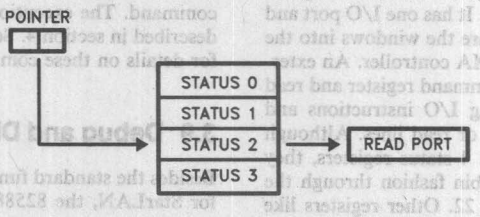


Figure 21. Register Access

4 Status registers are accessed through one read port



The pointer can be changed using a command or can be automatically incremented.

```
READ_STATUS_588: PROCEDURE;                    /* COMMAND 15 */
     OUTPUT (CS_588) = 15;                   /* RELEASE POINTER, INITIAL = 00 */
     STATUS_588(0)=INPUT (CS_588);           /* REFRESH STATUS REGISTER IMAGE */
     STATUS_588(1)=INPUT (CS_588);           /* IN MEMORY.
     STATUS_588(2)=INPUT (CS_588);
     STATUS_588(3)=INPUT (CS_588);
     RETURN
END READ_STATUS_588;
```

READING 4 STATUS REGISTERS

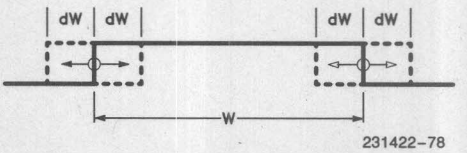
Figure 22. Reading the Status Register

nostics functions. The DIAGNOSE command of the 82588 does a self-test of most of the counters and timers in the 82588 serial unit. Using the DUMP command, all the internal registers of the 82588 can be dumped into the memory. The TDR command does Time Domain Reflectometery on the network. The 82588 has two loopback modes of operation. In the internal loopback mode, the TXD line is internally connected to the RXD one. No data appears outside the chip, and the 82588 is isolated from the link. This mode enables checking of the receive and transmit machines without link interference. In the external loopback mode, the 82588 becomes a full duplex device, being able to receive its own transmitted frames. In this mode data goes through the link and all CSMA/CD mechanisms are involved.

3.10 Jitter Performance

When the 82588 receives a frame from the HUB, the signal has jitter. Jitter is the shifting of the edges of the signal from their nominal position due to the transmission over a length of cable. Many factors like, intersymbol interference (pulses of different widths have different delays through the transmission media), rise and fall times of drivers and receivers, cross talk etc., contribute to the jitter. StarLAN specifies a maximum jitter of ±62.5 ns whenever the signal goes from a NODE/HUB or HUB/HUB. Figure 23 shows that the jitter tolerance of the 82588 is exactly the required ±62.5 ns at 1 Mbs for both 8X, 16X Manchester encoded data.

Jitter = ± variation of an edge from its nominal position.
Jitter can occur on every edge.



	x8	x16
Manchester	±1/16	±1/16
NRZI (Code Violations Enabled)	±1/16 BT	±3/32 BT
NRZI (Code Violations Disabled)	±3/16 BT	±3/16 BT

Figure 23. 82588 Jitter Performance

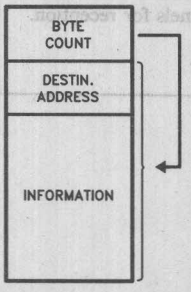
4.0 THE 82588

This chapter describes the basic 82588 operations. Please refer to the 82588 reference manual in Intel Microcommunications Handbook for a detailed description. Basic operations like transmitting a frame, receiving a frame, configuring the 82588 and dumping the register contents are discussed here to give a feel for how the 82588 works.

4.1 Transmit and Retransmit Operations

To transmit a frame, the CPU prepares a block in the memory called the transmit data block. As shown in Figure 24, this block starts with a byte count field, indicating how long the rest of the block is. The destination address field contains the node address of the destination. The rest of the block contains the information or the data field of the frame. The CPU also programs the DMA controller with the start address of the transmit data block. The DMA byte count must be equal to or greater than the block length. The 82588 is then issued a TRANSMIT command—an OUT instruction to the command port of the 82588. The 82588 starts generating DMA requests to read in the transmit data block by DMA. It also determines whether and how long it must defer on the link and after that, it starts transmitting the preamble. The 82588 constructs the frame on the fly. It takes the destination address from the memory, source address from its own individual address memory (previously programmed), data field from the memory and the CRC, is generated on chip, at the end of the frame.

1. Prepare Transmit Data—Block in Memory
2. Program DMA Controller
3. Issue Transmit Command on the Desired Channel



231422-25

Transmit Data Block

4. Interrupt is received on completion of command or if the command was aborted or there was a collision. The status bytes 1 and 2 indicate the result of the operation.

7	6	5	4	3	2	1	0	
TX DEF	HRT BEAT	MAX COLL		NUM. OF COLLISIONS				STATUS 1
COLL		TX OK			LOST CRS	LOST CTS	UNDER RUN	STATUS 2

231422-26

Transmit & Retransmit Results Format

Figure 24. Transmit Operation

At the conclusion of transmission the 82588 generates an interrupt to the CPU. The CPU can then read the status registers to find out if the transmission was successful. If a collision occurs during transmission, the 82588 aborts transmission and generates the jam sequence, as required by IEEE 802.3, and informs the CPU through interrupt and the status registers. It also starts the back-off algorithm.

To re-attempt transmission, the CPU must reinitialize the DMA controller 7to the start of the transmit data block and issue a RETRANSMIT command to the 82588. When the 82588 receives the retransmit command and the back-off timer has expired, it transmits again. Interrupt and the status register contents again indicate the success or failure of the (re)transmit attempt.

The main difference between transmit and retransmit commands is that retransmit does not clear the internal count for the number of collisions occurred, whereas transmit does. Moreoever, retransmit takes effect only when the back-off timer has expired.

4.2 Configuring the 82588

To initialize the 82588 and program its network and system parameters, a configure operation is performed. It is very similar to the transmit operation. Instead of a transmit data block as in transmit command, a configure data block—shown in Figure 12—is prepared by the CPU in the memory. The first two bytes of the block specify the length of the rest of the block, which specify the network and system parameters for the 82588. The DMA controller is then programmed by the CPU to the beginning of this block and a CONFIGURE command is issued to the 82588. The 82588 reads in the parameters by DMA and loads the parameters in the on-chip registers.

Similarly, for programming the INDIVIDUAL ADDRESS and MULTICAST ADDRESSes, the DMA controller is used to load the 82588 registers.

4.3 Frame Reception

Before enabling the 82588 for reception the CPU must make a buffer available for the frame to be received. The CPU must program the DMA controller with the starting address of the buffer and then issue the RX_ ENABLE command to the 82588. When a frame arrives at the RxD pin of the 82588, it starts being received. Only if the address in the destination address matches either the Individual address, Multicast address or if it is a broadcast address, is the frame deposited into memory by the 82588 using DMA. The format of storage in the memory is shown in Figure 25. At the end, a two byte field is attached which shows the status of the received frame. If CRC, alignment or overrun errors are encountered, they are reported. An inter-

1. Prepare a Buffer for Reception
2. Program DMA Controller
3. Issue Receiver Enable Command

When a frame is received, it is deposited in the memory. Receive status bytes (2) are appended to the frame in the memory, byte count written in the status registers 1, 2, and an interrupt is generated.

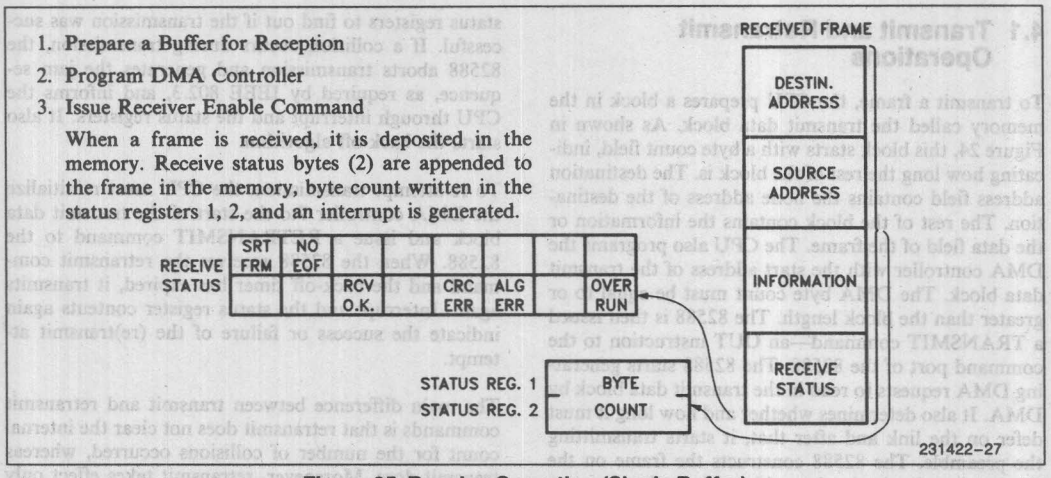


Figure 25. Receive Operation (Single Buffer)

rupt from 82588 occurs when all the bytes have been transferred to the memory. This informs the CPU that a new frame has been received.

If the received frame has errors, the CPU must recover (or re-use) the buffer. Note that the entire frame is deposited into one buffer. The 82588 when NOT configured for the external loopback mode, will detect collisions (code violations) during receptions. If a collision is detected, the reception is aborted and status updated. CPU is then informed by an interrupt (if the collided frame fragment is shorter than the address length, no reception will be started), and no interrupt will happen.

4.3.1 Multiple Buffer Frame Reception

It is also possible to receive a frame into a number of fixed size buffers. This is particularly economical if the received frames vary widely in size. If the single buffer scheme were used as described above, the buffer required would have to be bigger than the longest expected frame and would be very wasteful for very short (typically acknowledge or control) frames. The multiple buffer reception is illustrated in Figure 26. It uses two DMA channels for reception.

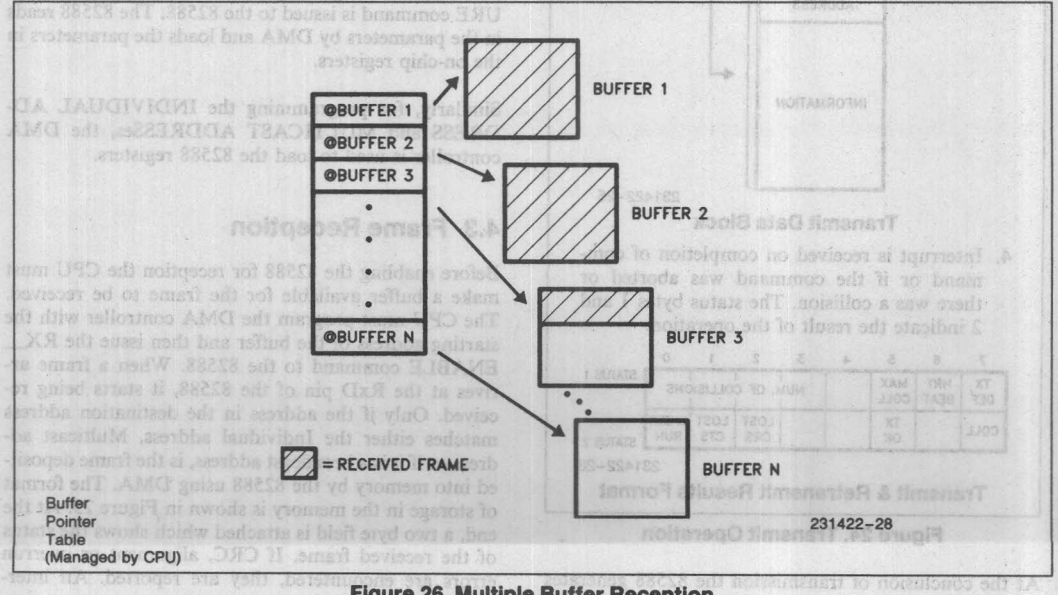


Figure 26. Multiple Buffer Reception

As in single buffer reception, the one channel, say channel 0, of the DMA controller is programmed to the start of buffer 1, and the 82588 is enabled for reception with the chaining bit set. As soon as the first byte is read out of the 82588 by the DMA controller and written into the first location of buffer 1, the 82588 generates an interrupt, saying that it is filling up its last available buffer and one more buffer must be allocated. The filling up of the buffer 1 continues. The CPU responds to the interrupt by programming the other DMA channel—channel 1—with the start address of the second buffer and issuing an ASSIGN ALTERNATE buffer command with an INTACK (interrupt acknowledge). This informs the 82588 that one more buffer is available on the other channel. When buffer 1 is filled up (the 82588 knows the size of buffers from the configuration command), the 82588 starts generating the DMA requests on the other channel. This automatically starts filling up buffer 2. As soon as the first byte is written into buffer 2, the 82588 interrupts the CPU again asking for one more buffer. The CPU programs the channel 0 of the DMA controller with the start address of buffer 3, issues an ASSIGN ALTERNATE buffer command with INTACK. This keeps the buffer 3 ready for the 82588. This switching of channels continues until the entire frame is received generating an end of frame interrupt. The CPU maintains the list of pointers to the buffers used.

Since a new buffer is allocated at the time of filling up of the last buffer, the 82588 automatically switches to the new buffer to receive the next frame as soon as the last frame is completely received. It can start receiving the new frame almost immediately, even before the end of frame interrupt is serviced and acknowledged by the CPU. If a new frame comes in, and the previous frame interrupt is not yet acknowledged, another interrupt needed for new buffer allocation is buffered (and not lost). As soon as the first one is acknowledged, the interrupt line goes active again for the buffered one.

If by the time a buffer fills up no new buffer is available, the 82588 keeps on receiving. An overrun will occur and will be reported in the received frame status. However, ample time is available for the allocation of a new buffer. It is roughly equal to the time to fill up a buffer. For 128 byte buffers it is $128 \times 8 = 1024$ ms or approximately 1 millisec. You get 1 ms to assign a new buffer after getting the interrupt for it. Hence the process of multiple buffer reception is not time critical for the system performance.

This method of reception is particularly useful to guarantee the reception of back-to-back frames separated by IFS time. This is because a new buffer is always available for the new frame after the current frame is received.

Although both the DMA channels get used up in receiving, only one channel is kept ready for reception and the other one can be used for other commands until the reception starts. If an execution command like transmit or dump command is being executed on a channel which must be allocated for reception, the command gets automatically aborted when the ASSIGN ALTERNATE BUFFER command is issued to the channel used for the execution command. The interrupt for command abortion occurs after the end of frame interrupt.

4.4 Memory Dump of Registers

All the 82588 internal registers can be dumped in the memory by the DUMP command. A DMA channel is used to transfer the register contents to the memory. It is very similar to reception of a frame; instead of data from the serial link, the data from the registers gets written into the memory. This provides a software debugging and diagnostic tool.

4.5 Other Operations

Other 82588 operations like DIAGNOSE, TDR, ABORT, etc. do not require any parameter or data transfer. They are executed by writing a command to the 82588 command register and knowing the results (if any) through the status registers.

5.0 StarLAN NODE FOR IBM PC

This chapter deals with the hardware—the StarLAN board—to interface the IBM PC to a StarLAN Network. This is a slave board which takes up one slot on the I/O channel of the IBM PC. Figure 27 shows an abstract block diagram of the board. It requires the IBM PC resources of the CPU, memory, DMA and interrupt controller on the system board to run it. Such a board has two interfaces. The IBM PC I/O Channel on the system or the parallel side and the telephone grade twisted pair wire on the serial side. Figures 28, 29 show the circuit diagram of the board.

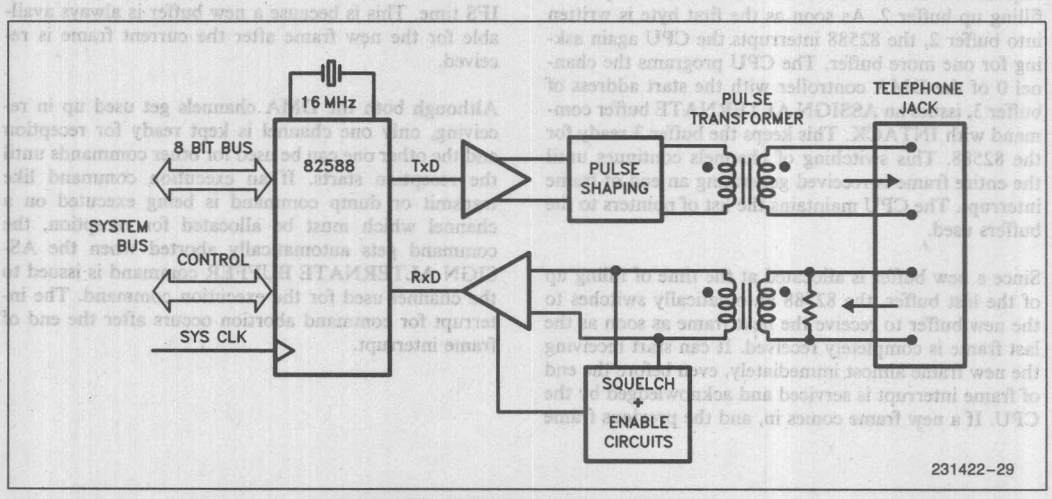


Figure 27. 82588 Based StarLAN Node

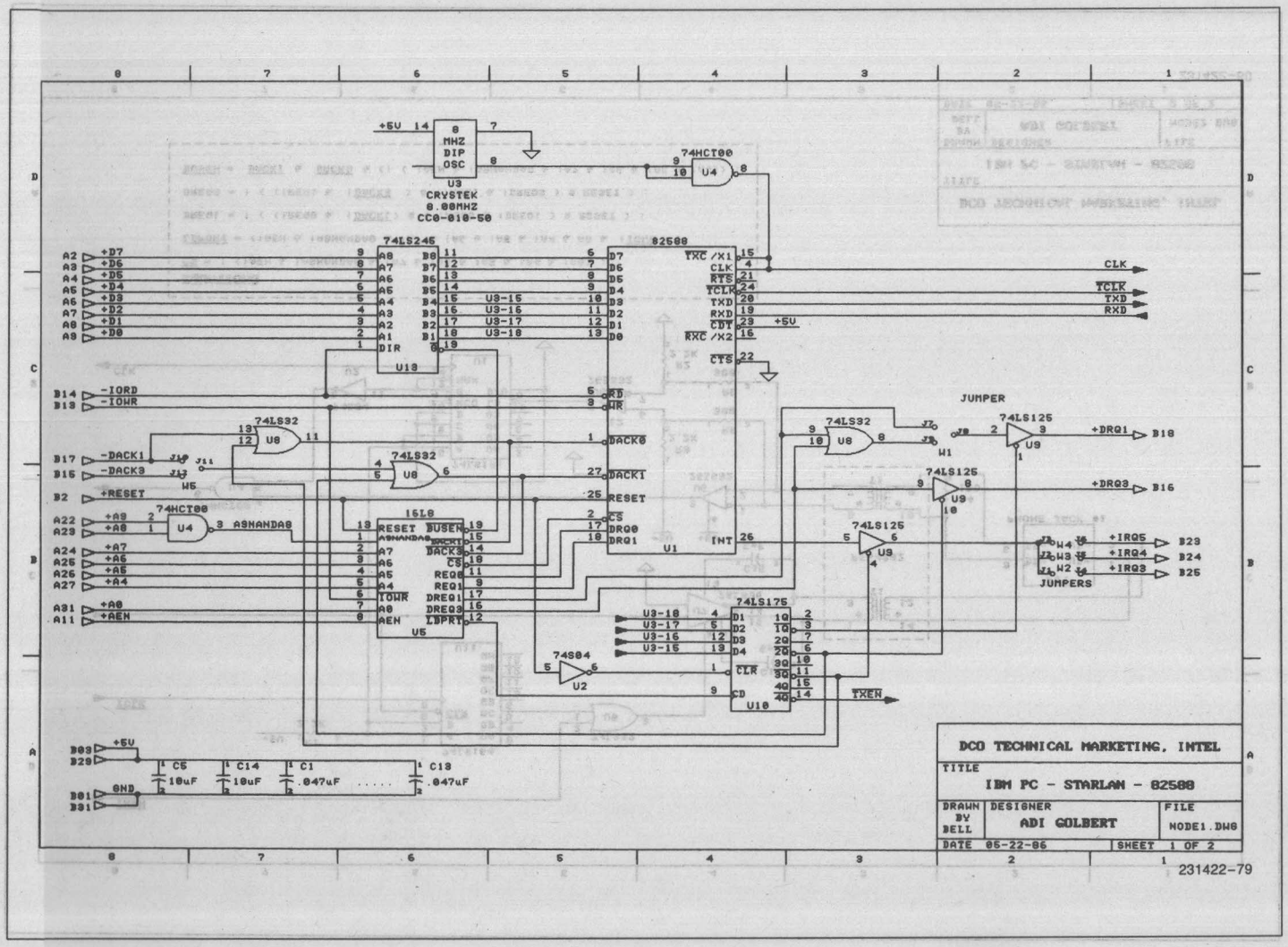


Figure 28
1-445

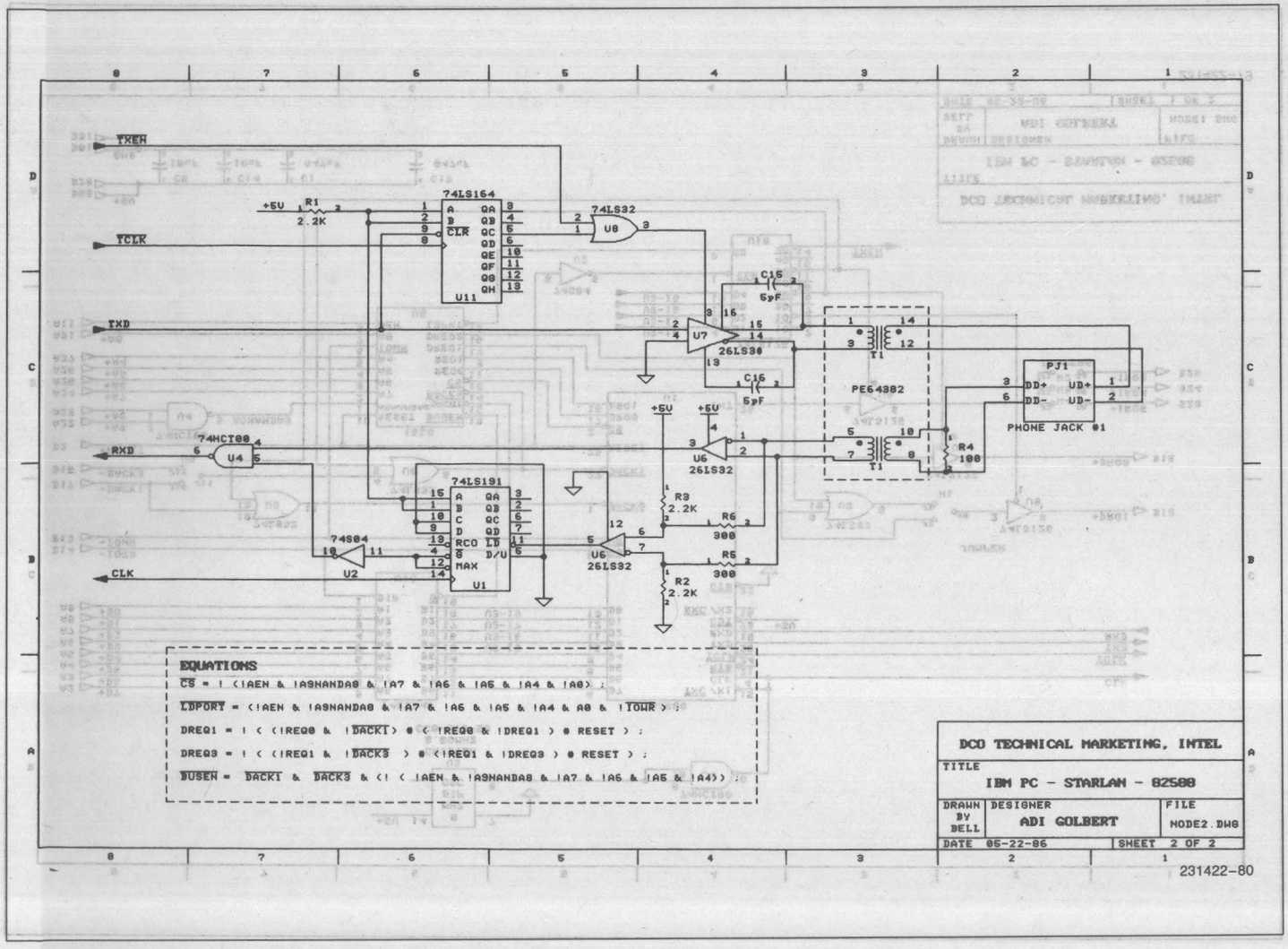

TXEN
TCLK
TXD
RXD
CLK
+5V
R1
2.2K
74LS164
U11
74LS32
U8
C15
5pF
U7
26LS30
T1
PE64382
PJ1
PHONE JACK #1
R4
100
74HCT00
U4
74LS191
U1
74S04
U2
U6
26LS32
R3
2.2K
R6
300
R5
300
R2
2.2K
EQUATIONS
CS = ! (!AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4 & !A0) ;
LDPORT = (!AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4 & A0 & !IOWR) ;
DREQ1 = ! ((!REQ0 & !DACK1) # (!REQ0 & !DREQ1) # RESET) ;
DREQ3 = ! ((!REQ1 & !DACK3) # (!REQ1 & !DREQ3) # RESET) ;
BUSEN = DACK1 & DACK3 & (! (!AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4)) ;
DCO TECHNICAL MARKETING, INTEL
TITLE
IBM PC - STARLAN - 82588
DRAWN BY BELL
DESIGNER
ADI GOLBERT
FILE
NODE2.DWG
DATE 05-22-86
SHEET 2 OF 2
231422-80

Figure 29

5.1 Interfacing to the IBM PC I/O Channel

IBM PC has 8 slots on the system board to allow expansion of the basic system. All of them are electrically identical and the I/O channel is the bus that links them all to the 8088 system bus. The I/O channel contains an 8 bit bidirectional data bus, 20 address lines, 6 levels of interrupt, 3 channels of DMA control lines and other control lines to do I/O and memory read/write operations. Figure 30 shows the signals and the pin assignment for the I/O Channel.

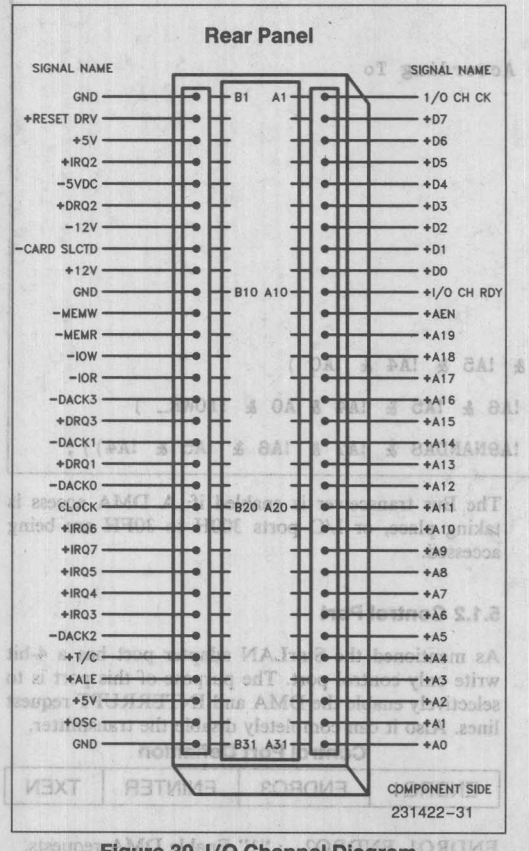


Figure 30. I/O Channel Diagram

5.1.1 REGISTER ACCESS AND DATA BUS INTERFACE

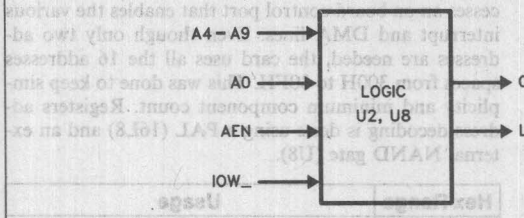
The CPU accesses the StarLAN adapter card through 2 I/O address windows. Address 300H is used to access to 82588 for commands and status, address 301H accesses an on board control port that enables the various interrupt and DMA lines. Even though only two addresses are needed, the card uses all the 16 addresses spaces from 300H to 30FH. This was done to keep simplicity and minimum component count. Registers address decoding is done using a PAL (16L8) and an external NAND gate (U8).

Hex Range	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Registers
0AX*	NMI Mask Register
0CX	Reserved
0EX	Reserved
200-20F	Game Control
210-217	Expansion Unit
220-24F	Reserved
278-27F	Reserved
2F0-2F7	Reserved
2F8-2FF	Asynchronous Communications (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Printer
380-38C**	SDLC Communications
380-389**	Binary Synchronous Communications (Secondary)
3A0-3A9	Binary Synchronous Communications (Primary)
3B0-3BF	IBM Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Asynchronous Communications (Primary)

* At power-on time, the Non Mask Interrupt into the 8088 is masked off.
This mask bit can be set and reset through system software as follows:
Set mask: Write hex 80 to I/O Address hex A0 (enable NMI)
Clear mask: Write hex 00 to I/O Address hex A0 (disable NMI)

** SDLC Communications and Secondary Binary Synchronous Communications cannot be used together because their hex addresses overlap.

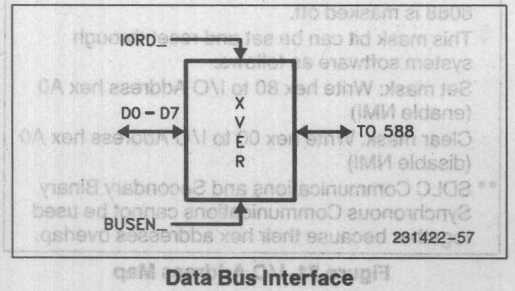
Figure 31. I/O Address Map



Register Access

```
Format of Following Equations Will Be According To

The Following Specifications:

    !  INVERT

    _  SIGNAL ACTIVE LOW

    &  LOGIC AND

    #  LOGIC OR

A9NANDA8 = ! (A9 & A8)

CS_ = ! ( !AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4 & !A0 )

LDPORT_ = ! ( !AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4 & A0 & !IOWR_ )

BUSEN_ = DACK1_ & DACK2_ & (! ( !AEN & !A9NANDA8 & !A7 & !A6 & !A5 & !A4));
```

The signal CS__ decodes address 300H, it is only active when AEN is inactive meaning CPU and not DMA cycles. LDPORT__ has exactly the same logic for address 301H, but it is only active during I/O write cycles. The I/O port sitting on address 301H is write only. The data BUS lines D0 to D7 are buffered from the 82588 to the PC bus using an 74LS245 transceiver chip.



Data Bus Interface

The Bus transceiver is enabled if: A DMA access is taking place, or I/O ports 300H to 30FH are being accessed.

5.1.2 Control Port

As mentioned the StarLAN adapter port has a 4-bit write only control port. The purpose of this port is to selectively enable the DMA and INTERRUPT request lines. Also it can completely disable the transmitter.

Control Port Definition

ENDRQ1	ENDRQ3	ENINTER	TXEN

ENDRQ1, ENDRQ2 : "1" Enable DMA requests.
ENINTER : "1" Enable INTERRUPT request.
TXEN : "1" Enable the transmitter.

On power up all bits default to "0".

5.1.3 CLOCK GENERATION

The 82588 requires two clocks for operation. The system clock and the serial clock. The serial clock can be generated on chip by putting a crystal across X1 and X2 pins. Alternatively, an externally generated clock can be fed in at pin X1 (with X2 left open). In both cases, the frequency must be either 8 or 16 times (sampling factor) the desired bit rate. For StarLAN, 8 or 16 MHz are the correct values to generate 1 Mb/s data rate. A configuration parameter is used to tell the 82588 what the sampling factor is. An externally supplied clock must have MOS levels (0.6V–3.9V). Specifications for the crystal and the circuit are shown in Figure 32.

The system clock has to be supplied externally. It can be up to 8 MHz. This clock runs the parallel side of the 82588. Its frequency does not have any impact on the read and write access times but on the rate at which data can be transferred to and from the 82588 (Maximum DMA data rate is one byte every two system clocks). This clock doesn't require MOS levels.

The I/O channel of the IBM PC supplies a 4.77 MHz signal of 33% duty cycle. This signal could be used as a system clock. It was decided, however, to generate a separate clock on the StarLAN board to be independent of the I/O channel clock so that this board can also be used in other IBM PCs and also in some other compatibles. The 8 MHz system clock is generated using a DIP OSCILLATOR which have the required 50 ppm tolerance to meet StarLAN. This clock is converted to MOS levels by 74HCT00 and fed into both the system and serial clock inputs.

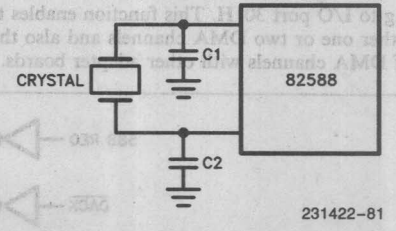
5.1.4 DMA INTERFACE

The 82588 requires either one or two DMA channels for full operation. In this application, one channel is dedicated for reception and the other is used for transmissions and the other commands. Use of only one DMA channel is possible but may require more complex software, also some RX frames may be lost during switches of the DMA channel from the receiver to the transmitter (Those frames will be recovered by higher layers of the protocol). Also using only one DMA channel will limit the 82588 loopback functionality. So the recommendation is to operate with two DMA channels if available. Appendix C describes a method of operating with only one DMA channel without loosing RX frames.

The IBM PC system board has one 8237A DMA controller. Channel 0 is used for doing the refresh of DRAMs. Channels 1, 2 and 3 are available for add-on boards on the I/O Channel. The floppy disk controller board uses the DMA channel 2 leaving exactly two channels (1 and 3) for the 82588. The situation is worse if the IBM PC/XT is used, since it uses channel 3 for the Winchester hard disk leaving just the channel 1 for

Series Resonance

—Frequency Will Drift by About 400 PPM from Nominal
—No Capacitors Needed
—Doesn't Meet StarLAN Requirements



Meeting StarLAN 100 PPM Requirements

—Use Parallel Resonance Crystal
—Recommended For Precise Frequencies
—82588 X-TAL Oscillator Stability ±35 PPM (0–70°C)

Crystal: Load Capacitance = 20 pF
Shunt Capacitance = 7 pF Maximum
Series Resistance = 30Ω Maximum
Frequency Tolerance = 50 PPM (0–70°C)

C1, C2 → 27 pF or 39 pF, 5%

Figure 32. Crystal Specifications

the 82588. On the other hand, the IBM PC/AT has 5 free DMA channels. We will assume that 8237A DMA channels 1 and 3 are available for the 82588 as in the case of the IBM PC.

Since the channel 0 of 8237A is used to do refresh of DRAMs all the channels should be operated in single byte transfer mode. In this mode, after every transfer for any channel the bus is granted to the current highest priority channel. In this way, no channel can hog the bus bandwidth and, more important, the refresh of DRAMs is assured every 15 microseconds since the refresh channel (number 0) has the highest priority. This mode of operation is very slow since the HOLD is dropped by the 8237A and then asserted again after every transfer. Demand mode of operation is a lot more suitable to 82588 but it cannot be used because of the refresh requirements.

Whenever the 82588 interfaces to the 8237A in the single transfer mode, there is a potential 8237A lock-up problem. The 82588 may deactivate its DMA request line (DREQ) before receiving an acknowledge from the DMA controller. This situation may happen during command abortions, or aborted receptions. The 8237A under those circumstances may lock-up. In order to solve this potential problem, an external logic must be used to insure that DREQ to the DMA controller is never deactivated before the acknowledge is received. Figure 33 shows the logic to implement this function. This logic is implemented in the 16L8 PAL.

The 82588 DREQ lines are connected to the IBM/PC bus through tri-state buffers which are enabled by writing to I/O port 301H. This function enables the use of either one or two DMA channels and also the sharing of DMA channels with other adapter boards.

5.1.5 INTERRUPT CONTROLLER

The 82588 interrupts the CPU after the execution of a command or on reception of a frame. It uses the 8259A interrupt controller on the system board to interrupt the CPU. There are 6 interrupt request lines, IRQ2 to IRQ7, on the I/O channel. Figure 34 shows the assignment of the lines. In fact, none of the lines are completely free for use. To add any new peripheral which uses a system board interrupt, this interrupt needs to have the capability to share the specific line, by driving the line with a tri-state driver. The 82588 StarLAN adapter board can optionally drive interrupt lines IRQ3, IRQ4 or IRQ5 (An 74LS125 driver is used).

Number	Usage
NMI	Parity
0	Timer
1	Keyboard
2	Reserved
3	Asynchronous Communications (Secondary) SDLC Communications BSC (Secondary)
4	Asynchronous Communications (Primary) SDLC Communications BSC (Primary)
5	Fixed Disk
6	Diskette
7	Printer

Figure 34. IBM PC Hardware Interrupt Listing

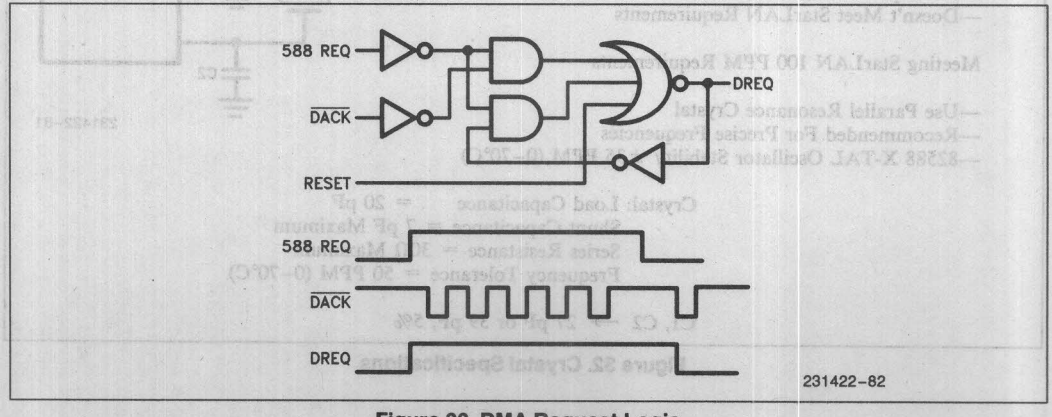


Figure 33. DMA Request Logic

5.2 Serial Link Interface

A typical StarLAN adapter board is connected to the twisted pair wiring using an extension cable (typically up to 8 meters—25 ft.). See Figure 35. One end of the cable plugs into the telephone modular jack on the StarLAN board and the other end into a modular jack in the wall. The twisted pair wiring starts at the modular jack in the wall and goes to the wiring closet. In the wiring closet, another telephone extension cable is used to connect to a StarLAN HUB. The transmitted signal from the 82588 reach the on-board telephone jack through a RS-422 driver with pulse shaping and a pulse transformer. The received signals from the telephone jack to the 82588 come through a pulse transformer, squelch circuit and a receive enable circuit.

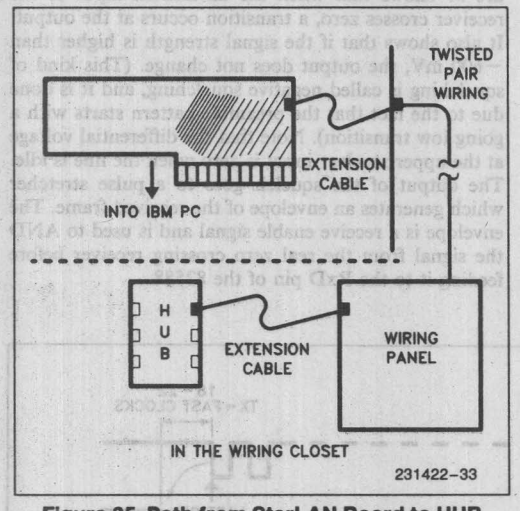


Figure 35. Path from StarLAN Board to HUB

5.2.1 TRANSMIT PATH

The single ended transmit signal on the TxD pin is converted to a differential signal and the rise and fall times are increased to 150 to 200 ns before feeding it to the pulse transformer (this pulse shaping is not a requirement, but proves to give good results). Am26LS30 is a RS-422 driver which converts the TxD signal to a differential signal. It also has slew rate control pins to increase to rise and fall times. A large rise and fall time reduces the possibility of crosstalk, interference and radiation. By the other hand a slower edge rate increases the jitter. In the StarLAN adapter card, the first approach was used. The 26LS30 converts a square pulse to a trapezoidal one—see Figure 36. The filtering effect of the cable further adds to reduce the higher frequency components from the waveform so that on the cable the signal is almost sinusoidal. The pulse transformer is for DC isolation. The pulse transformers from Pulse Engineering—type PE 64382—was used in this design. This is a dual transformer package which introduces an additional rise and fall time of about 70–100 ns on the signal, helping the former discussed waveshaping.

5.2.2 IDLE PATTERN GENERATION

StarLAN requires transmitters to generate an IDLE pattern after the last transmitted data bit. The IDLE pattern is defined to be a constant high level for 2–3 microseconds. The purpose of this pattern is to insure that receivers will decode properly the last transmitted data bits before signal decay. Currently the 82588 needs one external component to generate the IDLE. The operation principle is to have an external shift register (74LS164) that will kind of act as an envelope detector of the TXD line. Whenever the TXD line goes low

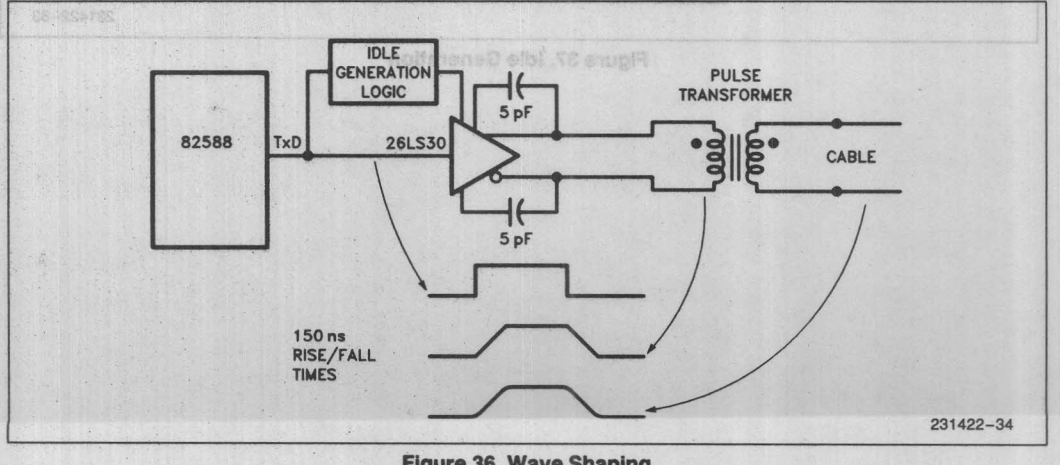


Figure 36. Wave Shaping

(first preamble bit), the output of the shift register (third cell) will immediately go low, enabling the RS-422 driver, the shift register being clocked by TCLK—will time the duration of the TXD high times. If the high time is more than 2 microseconds, meaning that the 82588 has gone idle, the transmitter will be disabled (See Figure 37). Another piece of this logic is the OR-ing of the output of the shift register with TXEN—signal which comes from the board control port. This signal completely disables the transmitter. The other purpose of this enable signal, is to make sure that after power-up, before the 82588 is configured, the RS-422 drivers won't be enabled (TCLK__ is not active before the configure command). See Figures 28, 29 for the complete circuit.

5.3 RECEIVE PATH

The signal coming from the HUB over the twisted pair wire is received on the StarLAN board through a 100Ω line termination resistor and a pulse transformer. The pulse transformer is of the same type as for the transmit side and its function is dc isolation. The received signal which is differential and almost sinusoidal is fed to the Am26LS32 RS-422 receiver. As seen from Figure 38 the pulse transformer feeds two RS-422 receivers. The one on the bottom is for squelch filtering and the one above is the real receiver which does real zero crossing detection on the signal and regenerates a square digital waveform from the sinusoidal signal that is received. Proper zero crossing detection is very essential; if the edges of the regenerated signal are not at zero crossings, the resulting signal may not be a proper Manchester encoded signal (self introduced jitter) even if the original signal is valid Manchester. The resistors in the lower receiver keep its differential inputs at a voltage difference of 600 mV. These bias resistors ensure that the output remains high as long as the input signal is more than −600 mV. It is very important that the RxD pin remains HIGH (not LOW or floating) whenever the receive line is idle. A violation of this may cause the 82588 to lock-up on transmitting. Remember, that based on the signal on the RxD pin, the 82588 extracts information on the data being received, Carrier Sense and Collision Detect. This squelch of 600 mV keeps the idle line noise from getting to the 82588. Figure 39 shows that when the differential input of the receiver crosses zero, a transition occurs at the output. It also shows that if the signal strength is higher than −600 mV, the output does not change. (This kind of squelching is called negative squelching, and it is done due to the fact that the preamble pattern starts with a going low transition). Note that the differential voltage at the upper receiver input is zero when the line is idle. The output of the squelch goes to a pulse stretcher which generates an envelope of the received frame. The envelope is a receive enable signal and is used to AND the signal from the real zero crossing receiver before feeding it to the RxD pin of the 82588.

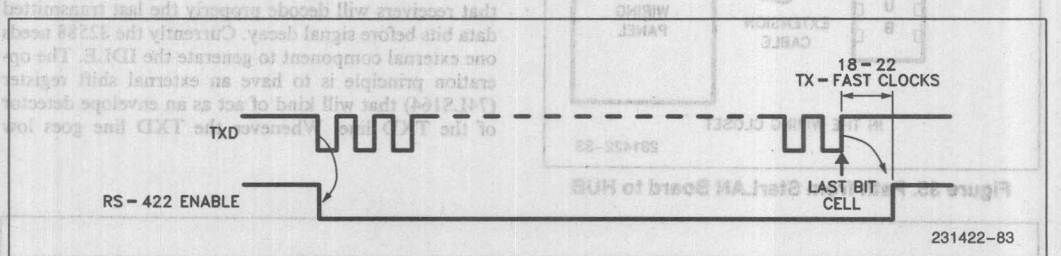


Figure 37. Idle Generation

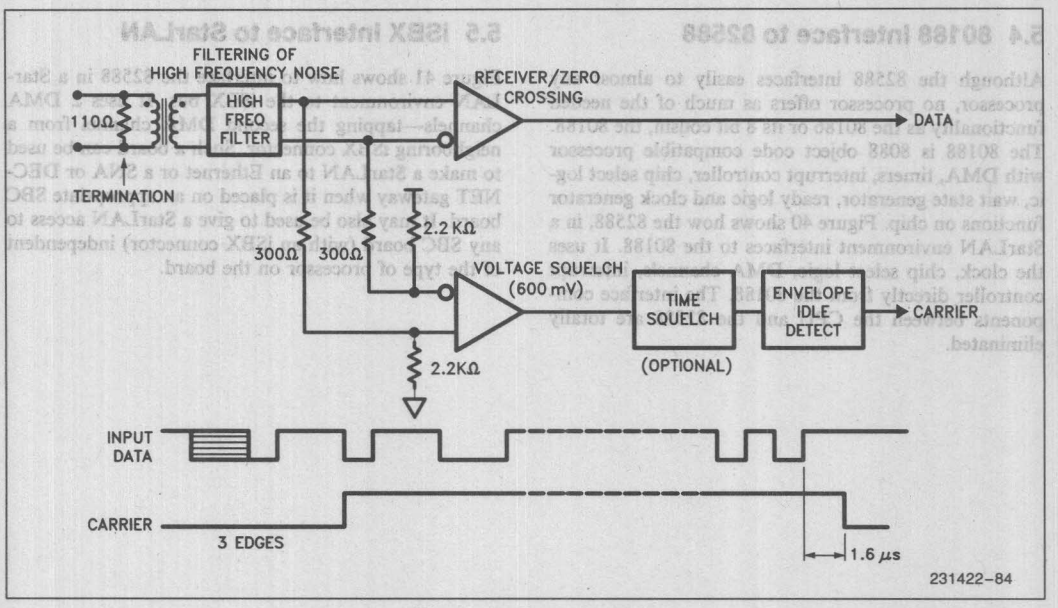


Figure 38. Input Ports

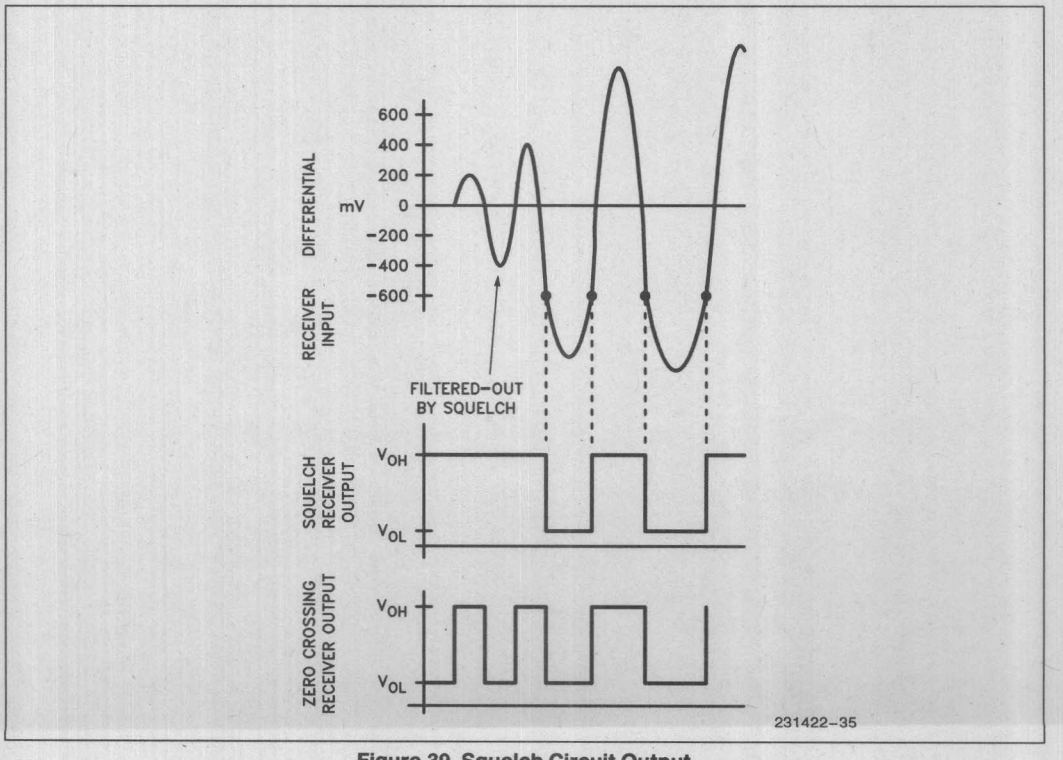


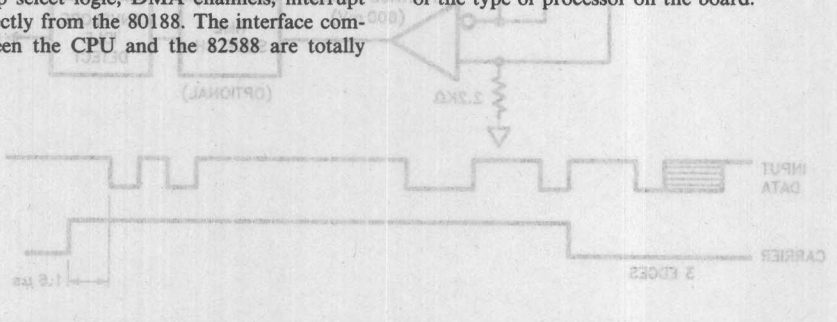
Figure 39. Squelch Circuit Output

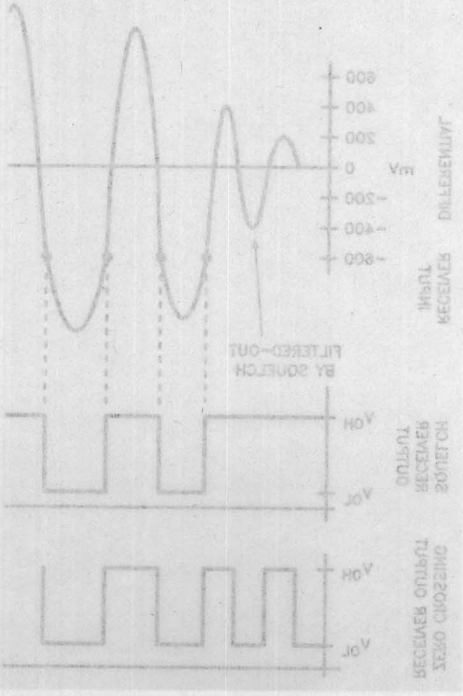
5.4 80188 Interface to 82588

Although the 82588 interfaces easily to almost any processor, no processor offers as much of the needed functionality as the 80186 or its 8 bit cousin, the 80188. The 80188 is 8088 object code compatible processor with DMA, timers, interrupt controller, chip select logic, wait state generator, ready logic and clock generator functions on chip. Figure 40 shows how the 82588, in a StarLAN environment interfaces to the 80188. It uses the clock, chip select logic, DMA channels, interrupt controller directly from the 80188. The interface components between the CPU and the 82588 are totally eliminated.

5.5 iSBX Interface to StarLAN

Figure 41 shows how to interface the 82588 in a StarLAN environment to the iSBX bus. It uses 2 DMA channels—tapping the second DMA channel from a neighboring iSBX connector. Such a board can be used to make a StarLAN to an Ethernet or a SNA or DECNET gateway when it is placed on an appropriate SBC board. It may also be used to give a StarLAN access to any SBC board (with an iSBX connector) independent of the type of processor on the board.





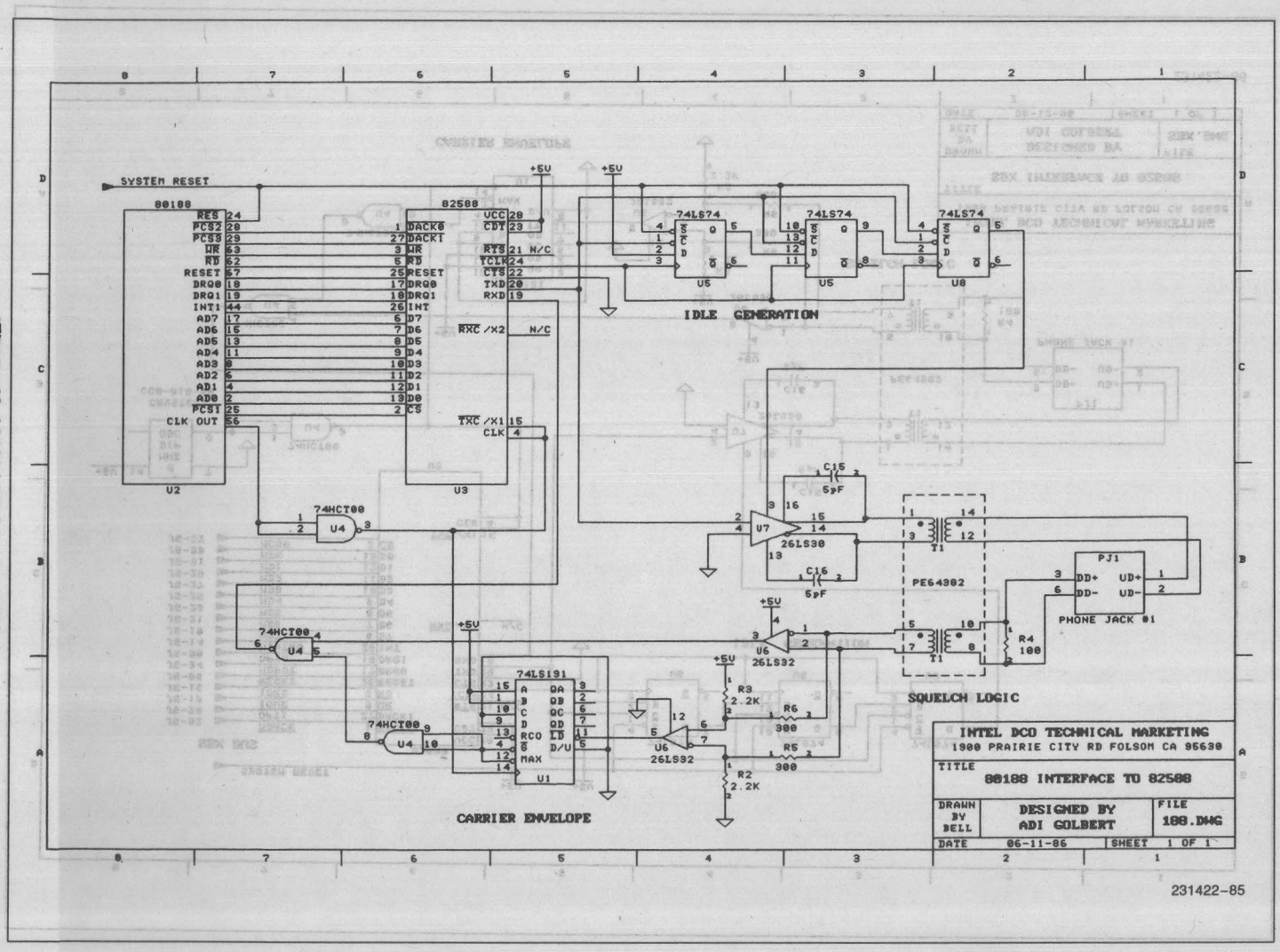


Figure 40. 80188 Interface to 82588

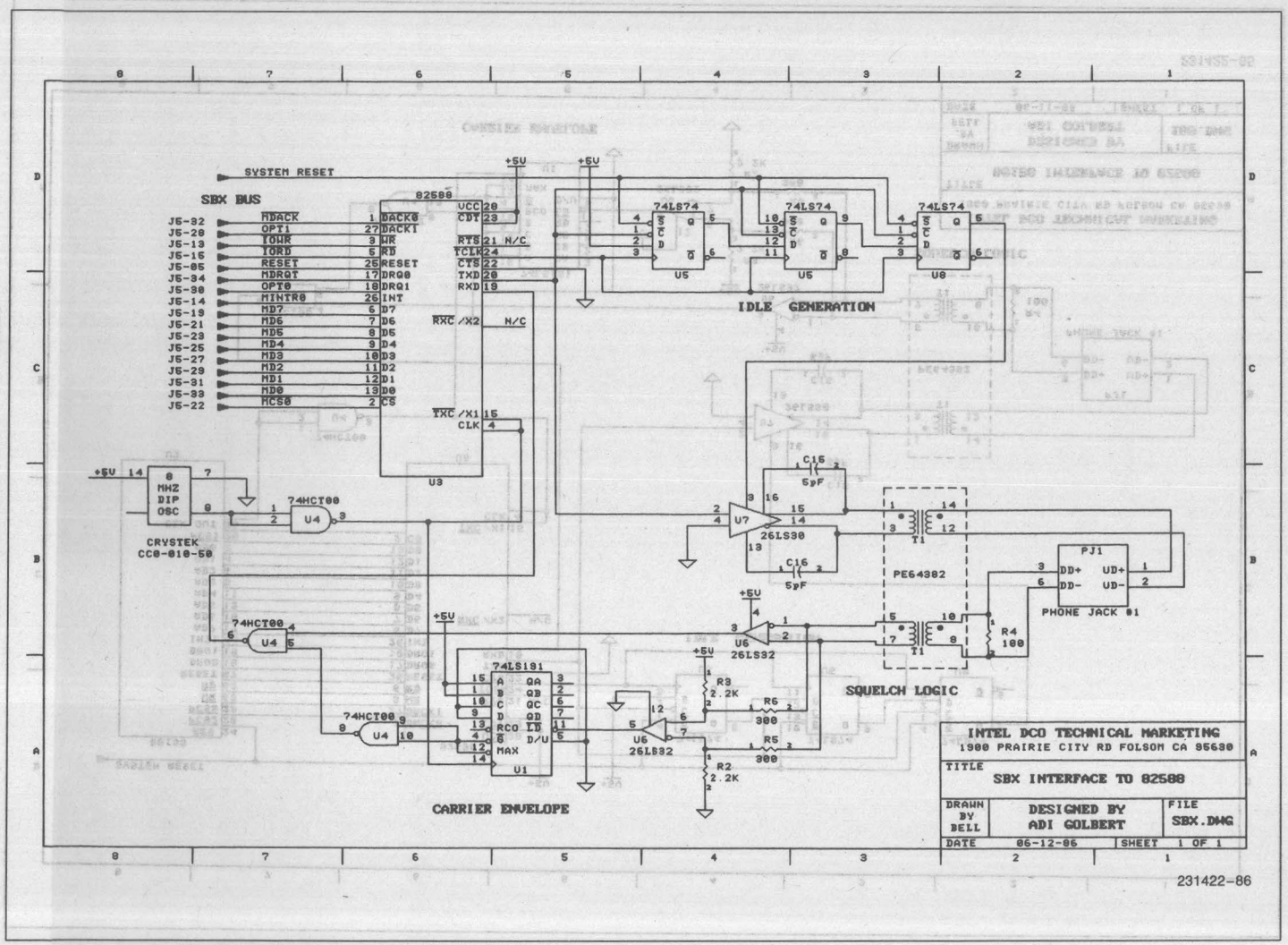


Figure 41. iSBX Interface to 82588

6.0 THE StarLAN HUB

The function of a StarLAN HUB is described in section 2.0. Figure 42 shows a block diagram of a HUB. It receives signals from the nodes (or lower level HUBs) detects if there is a collision, generates the collision presence signal, re-times the signal and sends it out to the higher level HUB. It also receives signals from the higher level HUB, re-times it and sends it to all the nodes and lower level HUBs connected to it. If there is no higher level HUB, a switch on the HUB routes the upstream received signal down to all the lower nodes. The functions performed by a HUB are:

*Receiving signals, squelch
*Carrier Sensing
*Collision Detection
*Collision Presence Signal Generation
*Signal Retiming
*Driving signals on to the cable
*Jabber Function
*Receive protection Timer

6.1 A StarLAN Hub for the IBM/PC

Figure 43 shows the implemention of a 5/6 port HUB for the IBM/PC.

The idea of the following design is to show a HUB that plugs into the IBM/PC backplane. This HUB not only gets its power from the backplane, but also enables the host PC to be one NODE into the StarLAN network. This embedded node scheme enables further savings due to the fact that all the analog interface for this port is saved (receiver, transmitter, transformer, etc).

This kind of board would suit very much a small cluster topology (very typical in departments and small offices) where the HUB board would be plugged into the FILE SERVER PC (PC/XT, PC/AT).

The HUB design doesn't implement the Jabber and the protection timers as called by the 1BASE5 draft standard. Those functions are optional and were not closed during the writing of this AP-NOTE. This HUB does implement the RETIMING circuit which is an essential requirement of StarLAN.

Figures 44 to 49 show a complete set of schematics for the HUB design.

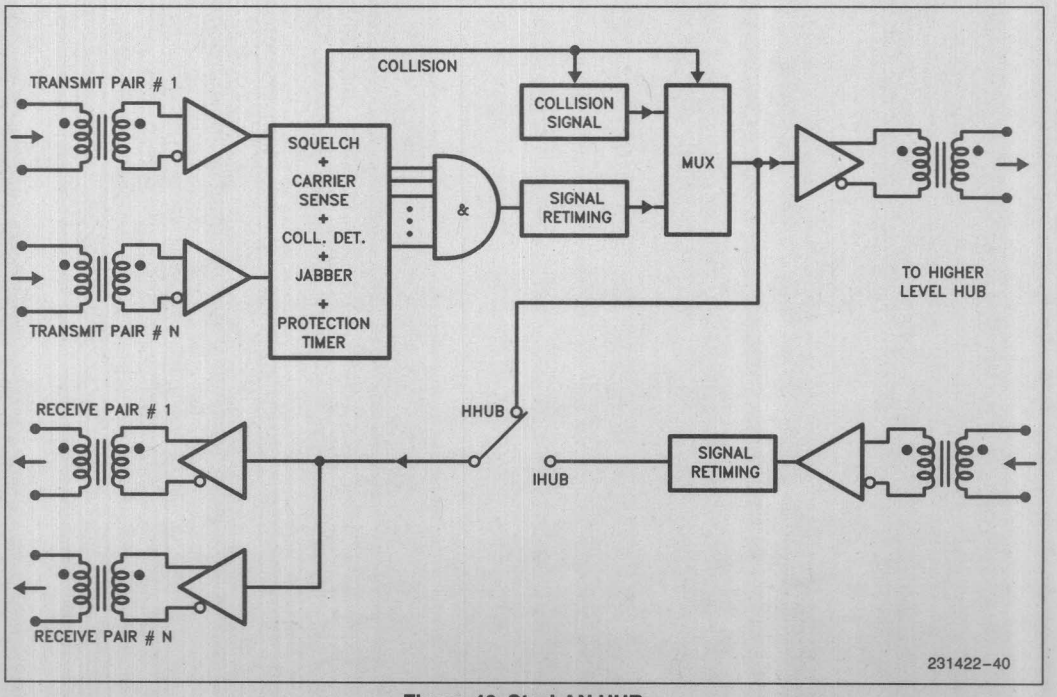


Figure 42. StarLAN HUB

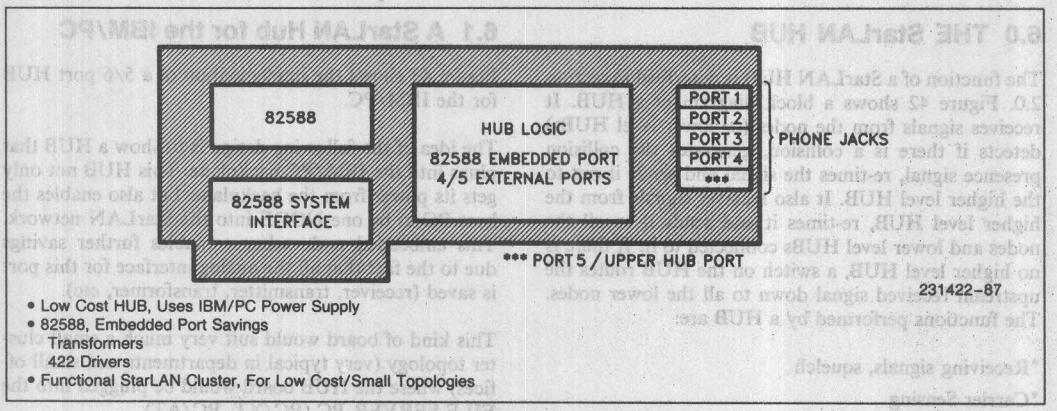
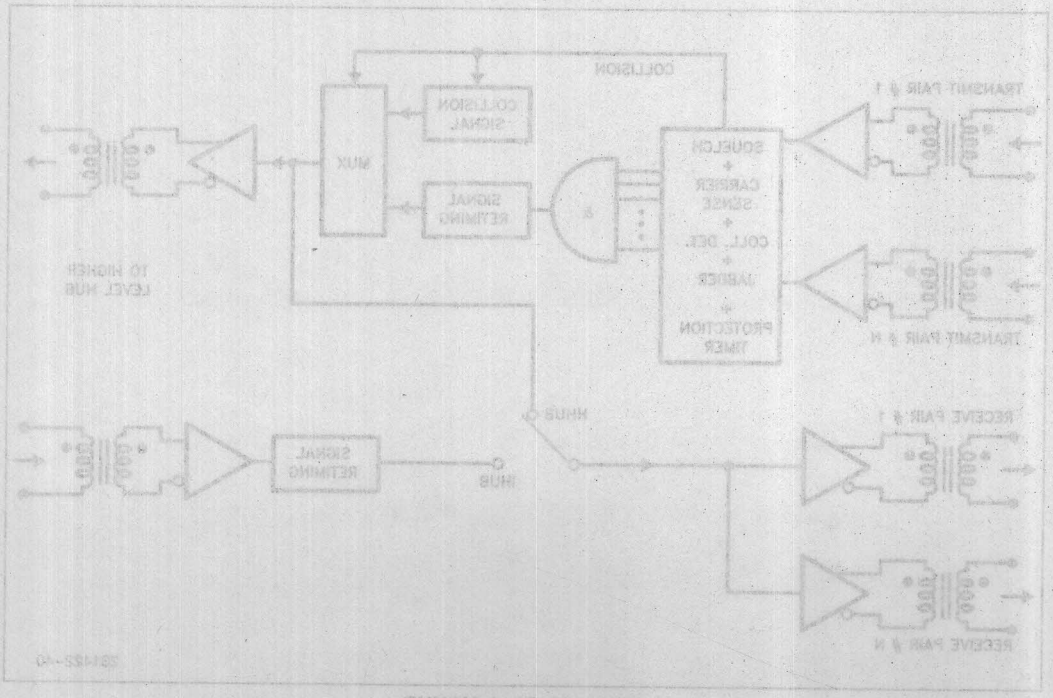


Figure 43. IBM/PC Resident HUB



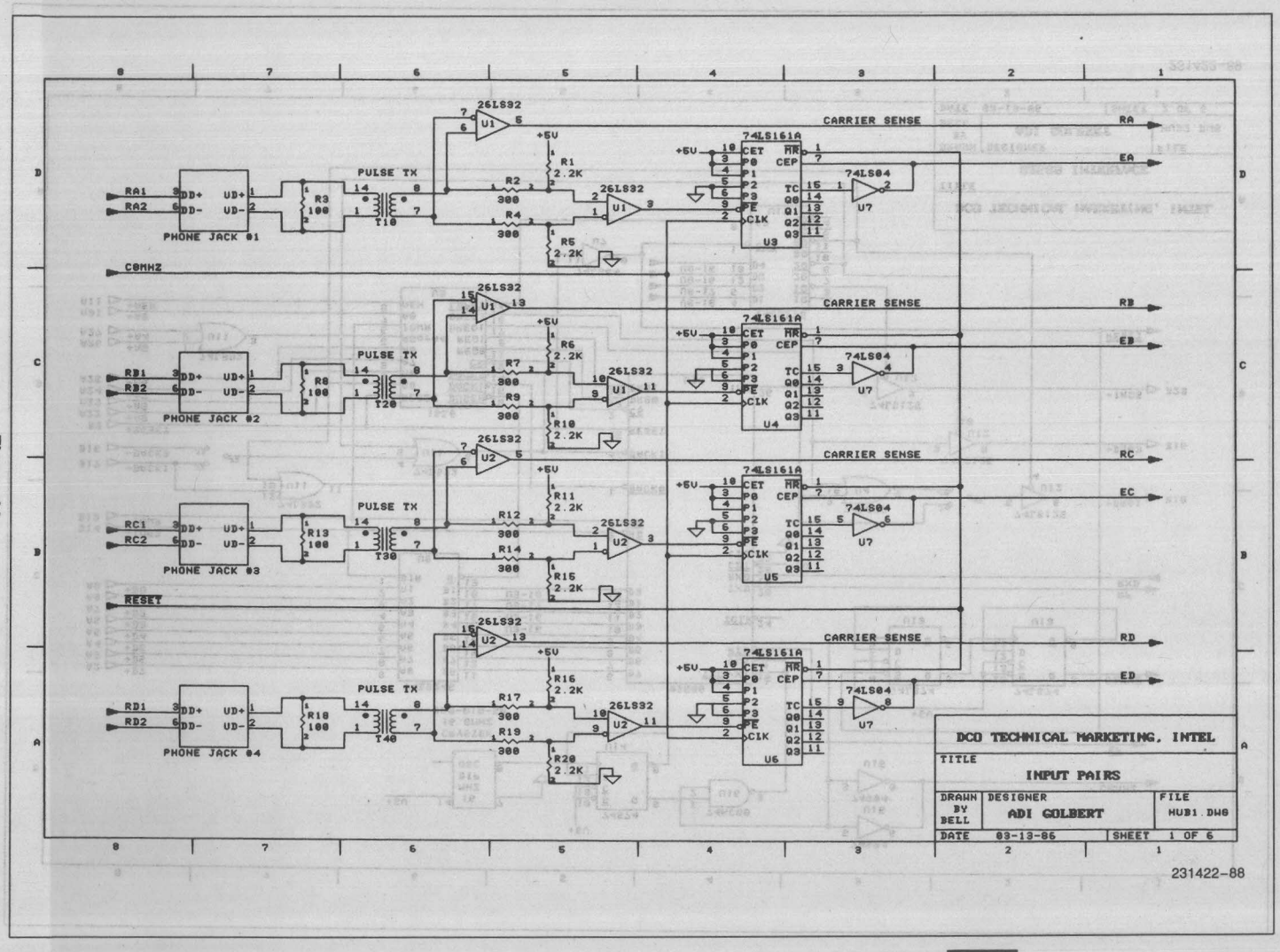

26LS32
74LS161A
74LS04
CARRIER SENSE
PULSE TX
PHONE JACK #1
PHONE JACK #2
PHONE JACK #3
PHONE JACK #4
+5V
COMMZ
RESET
RA
EA
RB
EB
RC
EC
RD
ED
DCD TECHNICAL MARKETING, INTEL
INPUT PAIRS
DESIGNER
ADI GOLBERT
HUB1.DWG
03-13-86
1 OF 6
231422-88

Figure 44

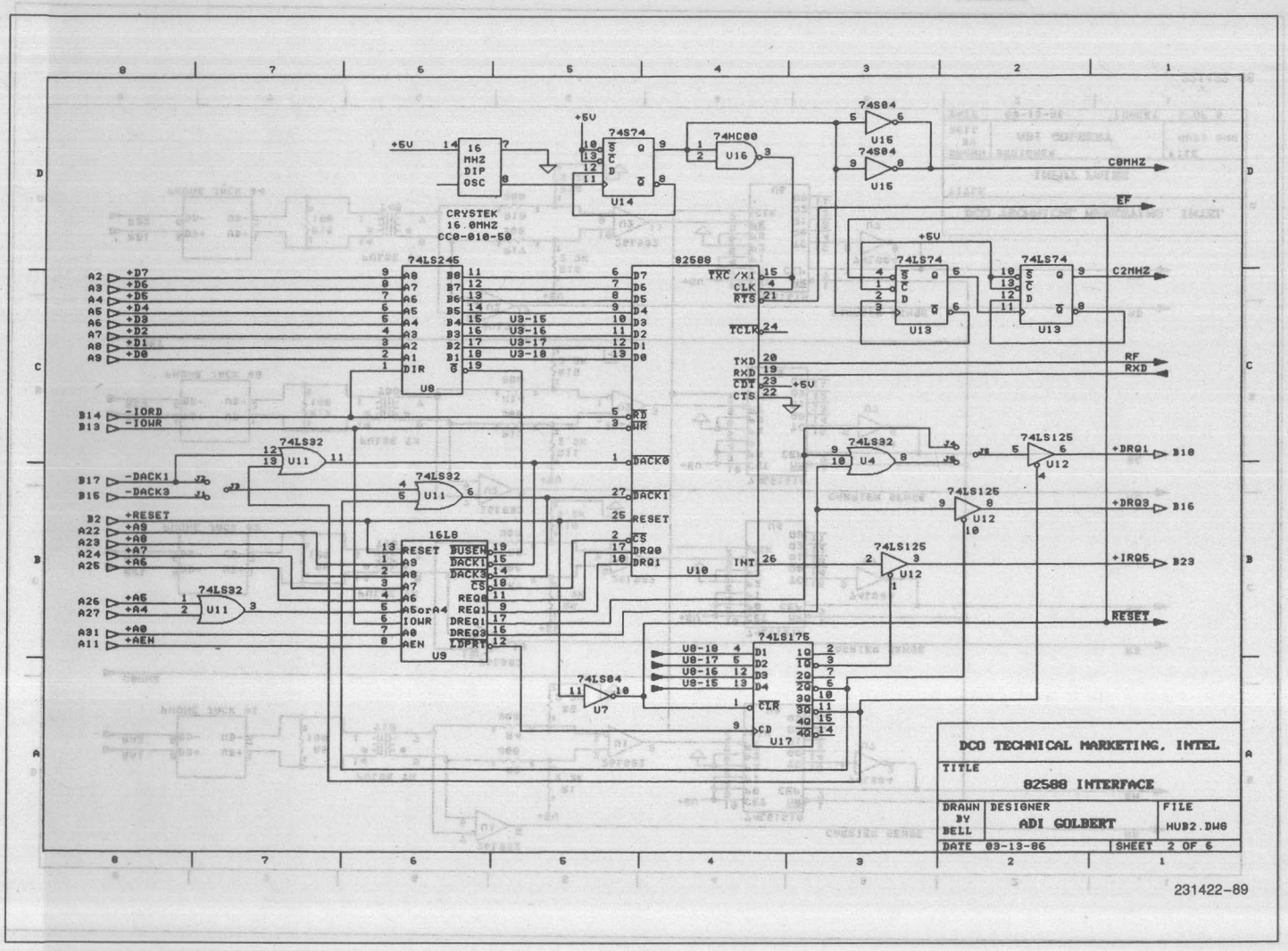

CRYSTEK
16.0MHZ
CC0-010-50
74LS245
82588
74LS32
16L8
74LS175
74LS125
74LS74
74S04
74HC00
DCO TECHNICAL MARKETING, INTEL
TITLE
82588 INTERFACE
DRAWN BY BELL
DESIGNER
ADI GOLBERT
FILE
HUB2.DW6
DATE 03-13-86
SHEET 2 OF 6
231422-89

Figure 45

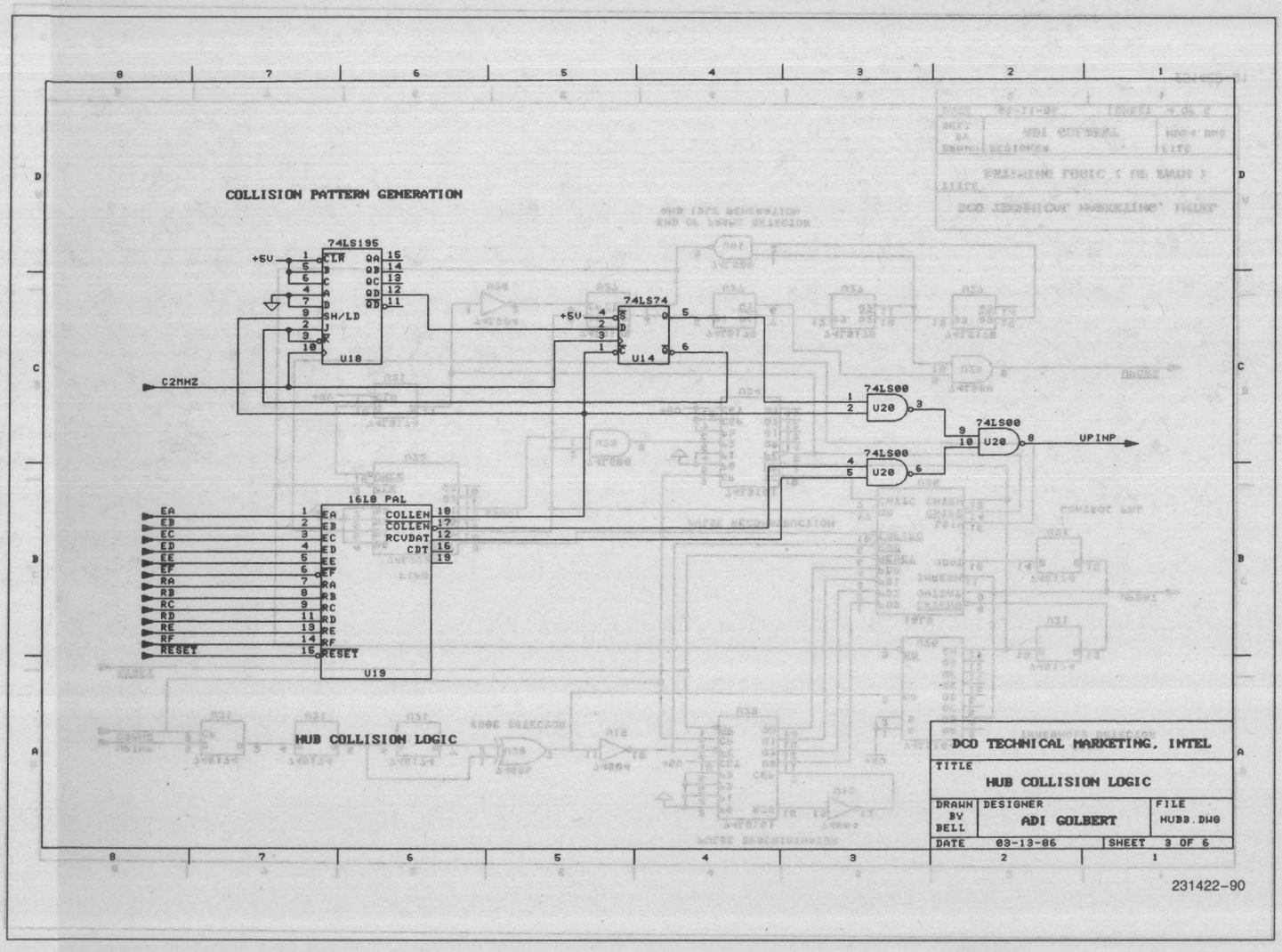


Figure 46
1-461

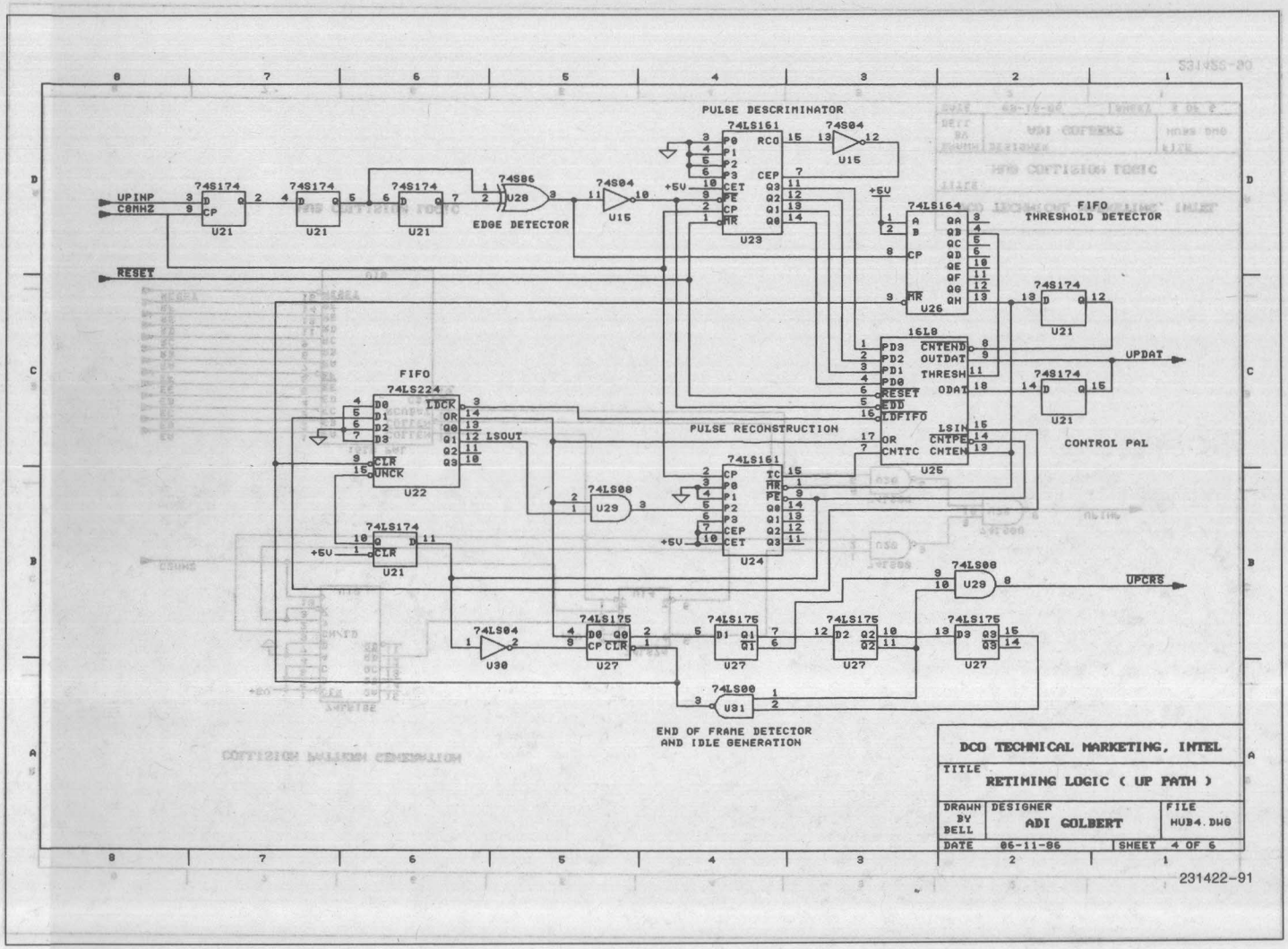

PULSE DESCRIMINATOR
74LS161
74S04
U15
UPINP
COMHZ
74S174
U21
74S86
U28
EDGE DETECTOR
74S04
U15
+5V
RESET
74LS164
FIFO
THRESHOLD DETECTOR
U26
74S174
U21
16L8
UPDAT
74S174
U21
CONTROL PAL
U25
FIFO
74LS224
U22
LSOUT
PULSE RECONSTRUCTION
74LS161
U24
74LS08
U29
74LS174
U21
74LS08
U29
UPCRS
74LS04
U30
74LS175
U27
74LS175
U27
74LS175
U27
74LS175
U27
74LS00
U31
END OF FRAME DETECTOR
AND IDLE GENERATION
DCO TECHNICAL MARKETING, INTEL
TITLE
RETIMING LOGIC (UP PATH)
DRAWN BY BELL
DESIGNER
ADI GOLBERT
FILE
HUB4.DWG
DATE
06-11-86
SHEET
4 OF 6
231422-91

Figure 47

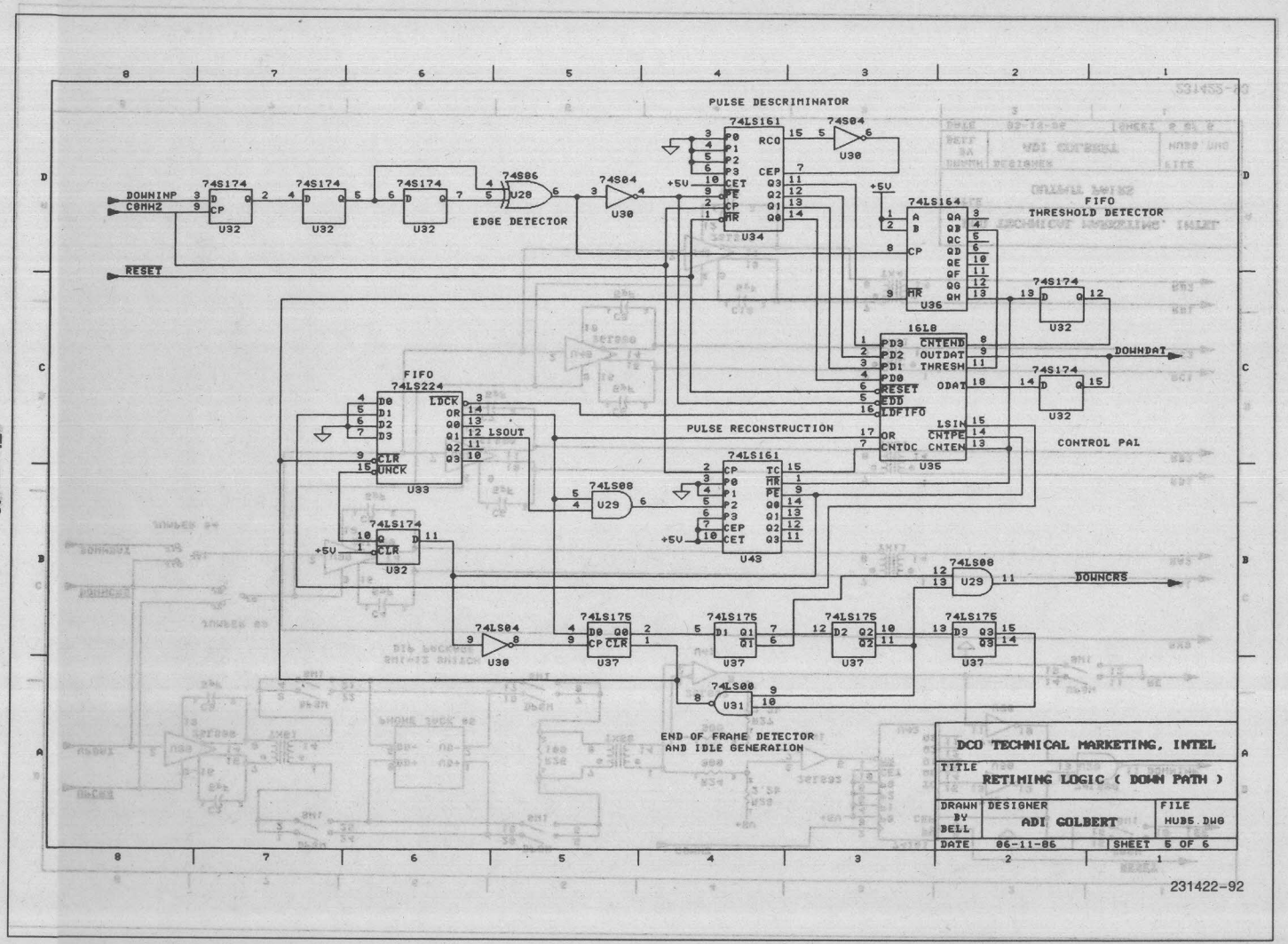

PULSE DESCRIMINATOR
74LS161
74S04
U30
DOWNINP
COMHZ
74S174
U32
74S174
U32
74S174
U32
74S86
U28
EDGE DETECTOR
74S04
U30
+5V
U34
+5V
74LS164
FIFO
THRESHOLD DETECTOR
U36
RESET
74S174
U32
16L8
DOWNDAT
74S174
U32
FIFO
74LS224
U33
LSOUT
PULSE RECONSTRUCTION
74LS161
U43
74LS08
U29
74LS174
U32
+5V
CONTROL PAL
U35
74LS08
U29
DOWNCRS
74S04
U30
74LS175
U37
74LS175
U37
74LS175
U37
74LS175
U37
74LS00
U31
END OF FRAME DETECTOR
AND IDLE GENERATION
DCO TECHNICAL MARKETING, INTEL
TITLE
RETIMING LOGIC (DOWN PATH)
DRAWN BY BELL
DESIGNER
ADI GOLBERT
FILE
HUB5.DWG
DATE
06-11-86
SHEET 5 OF 6
231422-92

Figure 48

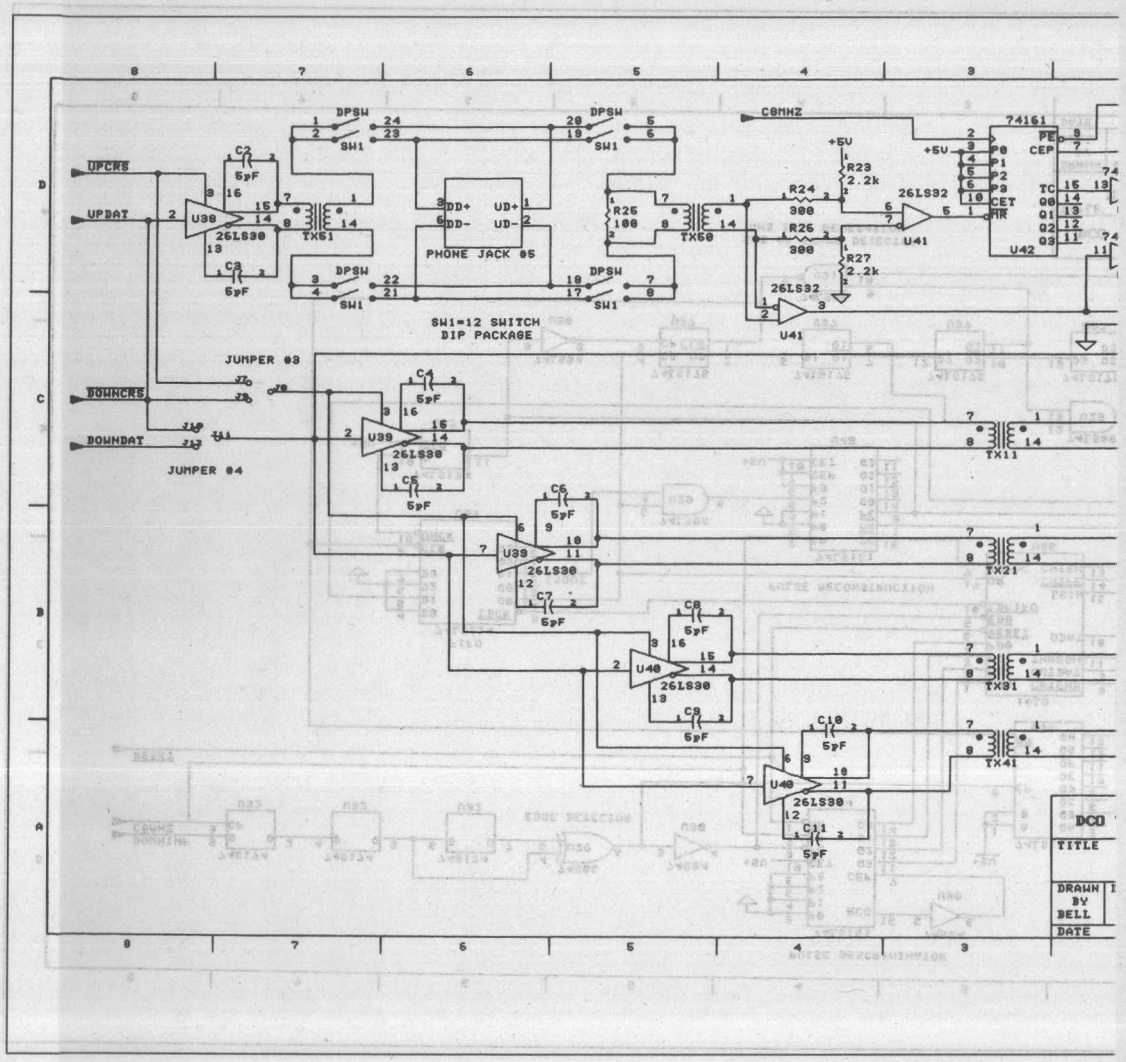

UPCNS
UPDAT
DOWNCNS
DOWNDAT
U38
U39
U40
26LS30
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
5pF
DPSW
SW1
SW1=12 SWITCH
DIP PACKAGE
PHONE JACK #5
DD+
DD-
UD+
UD-
TX51
TX60
TX11
TX21
TX31
TX41
R23
2.2k
R24
300
R25
100
R26
300
R27
2.2k
+5V
26LS32
U41
74161
U42
CEP
CET
PE
P0
P1
P2
P3
TC
Q0
Q1
Q2
Q3
CMHZ
JUMPER #3
JUMPER #4
J7
J8
J9
J10
J11
J12
DCO
TITLE
DRAWN
BY
BELL
DATE

Figure 49

6.1.1 HUB INPUT PORTS

Figure 38 shows a block diagram of an input port. Differently than the implementation in Figure 29 the HUB input port is potentially more complex than the NODE input port. The reason being that the HUB is a central resource and much more sensitive to noise. For example, if the NODE input port would falsely interpret noise on an IDLE line as valid signal, the worst case situation would be that this noise would be filtered out by the 82588 time squelch circuitry, on the HUB by the other hand, this false carrier sense could trigger a COLLISION and a good frame (on another input) potentially discarded.

As shown in Figure 38 immediately after the termination resistor, there is a HIGH FREQUENCY FILTER circuit. The purpose of this circuit is to eliminate high frequency noise components keeping noise jitter into the allocated budget (about ±30 ns). A 4 MHz two pole butterworth filter is being recommended by the IEEE 802.3 1BASE5 task force (see Figure 50).

The time squelch for the NODE board is implemented by the 82588 (see section 3.7) this circuit makes sure that pulses that are shorter than a specified duration will be filtered out.

The other components of the block diagram were explained in section 3.0.

The HUB design doesn't implement the HIGH FREQUENCY FILTER and TIME SQUELCH. In the HUB design as an output of each input port, two signals are available: Rn, En, (RA, RB . . . , EA, EB . . .). The Rn signals are the receive data after the zero crossing receivers. The En lines are CARRIER SENSE signals. The HUB design supports either 5 or 6 input ports, dependent upon if it is configured as IHUB or HHUB. Port RE, EE (Figure 49) is bidirectional, configurable for either input or output. Port RF, EF__ is the embedded 82588 port, and doesn't require the analog circuitry (EF is inverted, being generated from the RTS__ signal).

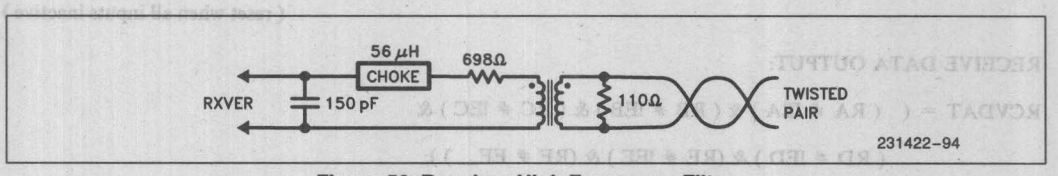


Figure 50. Receiver High Frequency Filter

6.1.2 COLLISION DETECTION

Rn and En signals from each channel are fed to a 16L8 PAL, where the collision detection function is performed.

Collision Detection in the StarLAN HUB is performed by detecting the presence of activity on more than one input channels. This means if the signal En is active for more than one channel, a collision is said to occur. This translates to the PAL equations:

COLLISION DETECTION:

```
CDT = ! (EA & !EB & !EC & !ED & !EE & EF__ #                    (only EA active)
         ! EA & EB & !EC & !ED & !EE & EF__ #                   (only EB active)
         ! EA & !EB & EC & !ED & !EE & EF__ #                   (only EC active)
         ! EA & !EB & !EC & ED & !EE & EF__ #                   (only ED active)
         ! EA & !EB & !EC & !ED & EE & EF__ #                   (only EE active)
         ! EA & !EB & !EC & !ED & !EE & !EF__ #                 (only EF active)
         ! EA & !EB & !EC & !ED & !EE & EF__ );                 (none of the inputs active)
```

COLLISION DETECTION SR-FF:

```
COLLEN__ = ! (CDT # COLLEN );                                   (set with collision)

COLLEN__ = ! ( RESET__ # COLLEN__ #
               ( !CDT & !EA & !EB & !EC & !ED & !EE & EF__);
                                                                ( reset when all inputs inactive )
```

RECEIVE DATA OUTPUT:

```
RCVDAT = (   ( RA # !EA ) & ( RB # !EB ) & ( RC # !EC ) &
             ( RD # !ED ) & (RE # !EE ) & (RF # EF__ ) );
                                                                ( output is high if no active input)
```

The COLLEN signal once triggered will stay active until all inputs go quiet. This signal is used externally to either enable passing RCVDAT or the collision presence signal (CPS) to the retiming logic. An external multiplexer using 3 nand gates is used for this function. Note that in this specific implementation the CPS/RCVDAT multiplexer is before the retiming logic, which is different from Figure 42 diagram. StarLAN provides enough BIT-BUDGET delay to allow the CPS signal to be generated through the retiming FIFO. In this HUB implementation it was decided to use this option to make sure that the CPS startup is synchronized with the previously transmitted bit as required by the 1BASE5 draft.

6.1.3 THE LOCAL 82588

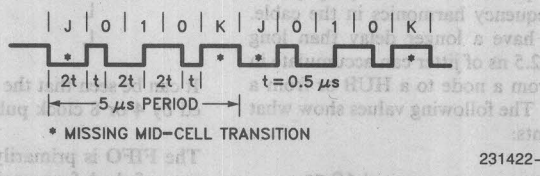
As described before, the purpose of the local 82588 is to enable the Host IBM/PC to also be a node into the StarLAN network. The interface of this 82588 is exactly similar to the one explained in section 5. The RTS__ signal serves as the carrier EF__ signal, and TXD as RF signal. This local node interfaces to the HUB without any analog interface which is a significant saving.

6.1.4 THE COLLISION PRESENCE SIGNAL

The Collision Presence Signal (CPS) is generated by the HUB whenever the HUB detects a collision. It then propagates the CPS to the higher level HUB. The CPS signal pattern is shown in Figure 51. Whenever a StarLAN node receives this signal, it should be able to detect within a very few bit times that a collision occurred. Since the nodes detect the occurrence of a collision by detecting violations in Manchester encoding, the CPS must obviously be a signal which violates Manchester encoding. Section 3.5 shows that the CPS has missing mid-cell transitions occurring every two and a half bit cells. These are detected as Manchester code violations. Thus, the StarLAN node is presented with collision detection indications every two and a half ms. This results in fast and reliable detection of collisions. CPS has a period of 5 ms.

One may wonder why such a strange looking signal was selected for CPS. The rationale is that this CPS looks very much like a valid Manchester signal—edges are 0.5 or 1.0 microsec. apart—resulting in identical radiation, cross-talk and jitter characteristics as a true Manchester. This also makes the re-timing logic for the signals simpler—it need not distinguish between valid Manchester and CPS. Moreover, this signal is easy to generate.

A few important requirements for CPS signal are: a) it should be generated starting synchronized with the last transmitted bit cell. CPS is allowed to start either low or high, but no bit cell of more than 1 microsecond is allowed (Avoid false idles, very long "low" bits). b) once it starts, it should continue until all the input lines to the HUB die out. Typically, when the collision occurs, the multiplexor in the HUB switches from RCV signal to the CPS. This switch is completely asynchronous to the currently being transmitted data, and by such may violate the requirement of not having bit cells longer than 1 μs. In order to avoid those long pulses, the output of the CPS/RCVDAT multiplexer is passed through the retiming circuitry which will correct those long pulses to their nominal value. The reason for restriction b) is to ensure that the CPS is seen by all nodes on the network since it is generated until every node has finished generating the Jam pattern.



- Collision Presence Signal (CPS) is generated by the HUB when it detects more than one input line active.
- CPS violates Manchester encoding rules—due to missing mid-cell transitions—hence is detected as a collision by the DTE (82588).

Choice of Collision Presence Signal

- It is a Manchester look-alike signal—edges are 0.5 or 1.0 μs apart.
  - — Identical radiation, crosstalk and jitter characteristics
  - — Eases retiming of the signal in the HUB
- It is easy to generate—1.5 TTL pack, or in a PAL

Figure 51. Collision Presence Signal

CPS is generated using a 4-bit shift register and a flip-flop as shown in Figure 52. It works off a 2 MHz clock. A closer look at the CPS waveform shows that it is inverse symmetric within the 5 μs period. The circuit is a 5-bit shift register with a complementary feedback from the last to the first bit. The bits remain in defined states (01100) till collision occurs. On collision the bits start rotating around generating the pattern of 0011011001, 0011011001, 00110 ... with each state lasting for 0.5 μs.

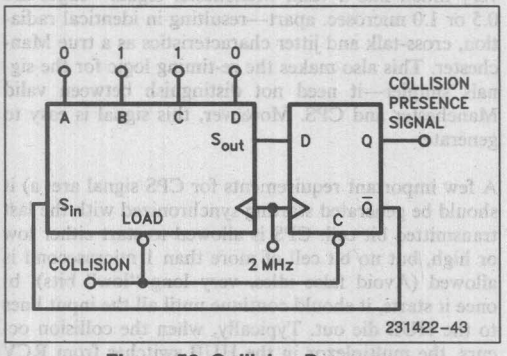


Figure 52. Collision Presence Signal Generation

6.1.5 SIGNAL RETIMING

Whenever the signal goes over a cable it suffers jitter. This means that the edges are no longer separated by the same 0.5 or 1.0 μs as at the point of origin. There are various causes of jitter. Drivers, receivers introduce some shifting of edges because of differing rise and fall times and thresholds. A random sequence of bits also produces a jitter which is called intersymbol interference, which is a consequence of different propagation delays for different frequency harmonics in the cable. Meaning short pulses have a longer delay than long ones. A maximum of 62.5 ns of jitter can accumulate in a StarLAN network from a node to a HUB or from a HUB to another HUB. The following values show what are the jitter components:

Transmitter skew	±10 ns
Cable Intersymbol interference	±9 ns
Cable Reflections	±8 ns
Reflections due to receiver termination mismatch	±5 ns
HUB fan-in, fan-out	±5 ns
Noise	±25.5
Total	±62.5 ns

It is important for the signal to be cleaned up of this jitter before it is sent on the next stretch of cable because if too much jitter accumulates, the signal is no longer meaningful. A valid Manchester signal would, as a result of jitter, may no longer be decodable. The process of either re-aligning the edges or reconstructing the signal or even re-generating the signal so that it once again "looks new" is called re-timing. StarLAN requires for the signal to be re-timed after it has travelled on a segment of cable. In a typical HUB two re-timing circuits are necessary; one for the signals going upstream towards the higher level HUB and the other for signals going downstream towards the nodes.

6.1.6 RETIMING CIRCUIT, THEORY OF OPERATION

This section will discuss the principles of designing a re-timing circuit. Figure 53 shows the block diagram of a re-timing circuit. The data coming in is synchronized using an 8 MHz sampling clock. Edges in the waveform are detected doing an XOR of two consecutive samples. A counter counts the number of 8 MHz clocks between two edges. This gives an indication of long (6 to 10 clocks) or short (3 to 5 clocks) pulses in the received waveform. Pulses shorter than 3 clocks are filtered out. Every time an edge occurs, the length—(S)hort or (L)ong—of the pulse is fed into the FIFO. Retiming of the waveform is done by actually generating a new waveform based on the information being pumped into the FIFO. The signal regeneration unit reads the FIFO and generates the output waveform out of 8 MHz clock pulses based on what it reads, either short or longs. In summary every time a bit is read from the fifo, it indicates that a transition needs to occur, and when to fetch the next bit. When idle the output of the retiming logic starts with a "high" level.

FIFO	Output
empty	1111
S	0000
S	1111
L	00000000
L	11111111

It can be seen that the output always has edges separated by 4 or 8 clock pulses—0.5 or 1.0 μs.

The FIFO is primarily needed to account for a difference of clock frequencies at the source and regeneration end. Due to this difference, data can come in faster or slower than the regeneration circuit expects. A 16 deep FIFO can handle frequency deviations of up to 200 ppm for frame lengths up to 1600 bytes. The FIFO also overcomes short term variations in edge separation. It is essential that the FIFO fills in up to about half before the process of regeneration is started. Thus, if the regeneration is done at a clock slightly faster than the source clock, there is always data in the FIFO to work from. That is why the FIFO threshold detect logic is necessary, which counts 8 edges and then enables the signal regeneration logic.

Example:

Input Waveform ...1111000111100000001111111111000111100...

Input into the FIFO

<S> <S> <L> <L><S> <S>

Regenerated Output:

Output: ...1111100001111000000001111111110000111...

FIFO:

<S> <S> <L> <L> <S> <S>

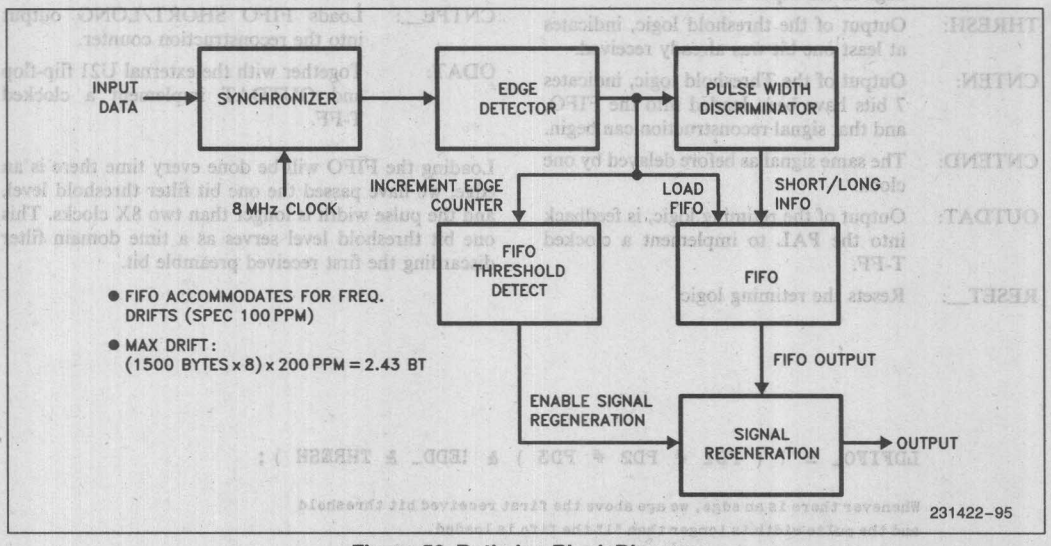


Figure 53. Retiming Block Diagram

6.1.7 RETIMING CIRCUIT IMPLEMENTATION

The retiming circuit implementation can be seen in Figures 47, 48. Both figures implement exactly the same function, one for the upstream, and the other for the downstream. The retiming circuit was implemented using about 8 SSI, MSI TTL components, one fifo chip and one PAL. The purpose of implementing this function with discrete components was to show the implementation details. The discussion of the implementation will refer to Figure 47 for unit numbers.

The signal UPIMP which is an output of the HUB multiplexing logic, is asynchronous to the local clock. This signal is synchronized by two flip-flops and fed into an edge generation logic (basically an XOR gate that compares the present sample with the previous one). On every input transition a 125 ns pulse will be generated at the output of the edge detector (U28). This pulse will reset the 74LS161 counter that is responsible for measuring pulse widths (in X8 clock increments). The output of the pulse discriminator will reflect the previous pulse width every time a new edge is detected. The following events will take place on every detected edge:

1. U26 which is the threshold detector will shift one "1" in. The outputs of U26 will be used by the control PAL to start the reconstruction process.
2. The output of U23 which specifies the last pulse width will be input into the control PAL for determining if it was a long or short pulse. The result of this evaluation will be the LSIN signal which will be loaded into the fifo (U22).

U22 is the retiming FIFO, it is 16x4 fifo, but only one bit is necessary to store the SHORT/LONG information.

CONTROL LOGIC PAL functions (U25):

Signals definition:

INPUTS:

PD0..PD3: Outputs of the pulse descriminator, indicate the width of the last measured pulse.

EDD__: Output of the edge detector, pulse of 125 ns width, indicates the occurrence of an edge in the input data.

THRESH: Output of the threshold logic, indicates at least one bit was already received.

CNTEN: Output of the Threshold logic, indicates 7 bits have been loaded into the FIFO, and that signal reconstruction can begin.

CNTEND: The same signal as before delayed by one clock.

OUTDAT: Output of the retiming logic, is feedback into the PAL to implement a clocked T-FF.

RESET__: Resets the retiming logic.

CNTTC: Terminal count of the reconstruction counter, indicating that reconstruction of a new bit will get started.

OR: Output of the FIFO indicating, that the FIFO is empty and that IDLE generation can get started.

OUTPUTS:

LDFIFO__: Loads SHORT/LONG indications into the FIFO.

LSIN: Indicates SHORT/LONG

CNTPE__: Loads FIFO SHORT/LONG output into the reconstruction counter.

ODAT: Together with the external U21 flip-flop and OUTDAT implement a clocked T-FF.

Loading the FIFO will be done every time there is an edge, we have passed the one bit filter threshold level, and the pulse width is longer than two 8X clocks. This one bit threshold level serves as a time domain filter discarding the first received preamble bit.

```
LDFIFO_ = ! ( PD1 # PD2 # PD3 ) & !EDD_ & THRESH );

Whenever there is an edge, we are above the first received bit threshold
and the pulse width is longer than "1" the fifo is loaded.

LSIN    = ! (PD3 # (PD2 & PD0) # (PD2 & PD1));

Every pulse longer than 6 is considered to be a long pulse.

CNTPE_  = ! ( (CNTEN & !CNTEND) # CNTTC );

The reconstruction counter is loaded in two conditions:

Whenever CNTEN comes active, meaning the FIFO threshold of seven was exceeded.
Whenever the terminal count of U24 is active meaning a new pulse is going to be reconstructed.

ODAT    = !RESET_ # (!CNTPE_ & !OUTDAT)    (A)
                  # ( CNTPE_ &  OUTDAT)    (B)
                  # (!CNTPE_ & !OR)        (C)

Minterm (A) and (B) implement a T-FF, whenever CNTPE_ is "low"
ODAT will toggle. The external U21 is part of this flip-flop.
Minterm (C) insures the output of the flip-flop will go inactive
"high" when the FIFO is empty. RESET_ causes the output to go
"high" on initialization.
```

U24 as mentioned is the reconstruction counter. This counter is loaded by the control logic with either 8 or 12, it counts up and is reloaded on terminal count. Essentially generating at the output nominal length longs and shorts.

U22 is the retiming FIFO, and its function as mentioned is to accommodate frequency skews between the incoming and outgoing signal.

U27 is the IDLE generation logic. The purpose of this logic is to detect when the FIFO is empty, meaning that no more data needs to be transmitted. On detection of this event this component will generate 2 ms of IDLE time. On the end of IDLE the whole retiming logic will be reset.

6.1.8 DRIVER CIRCUITS

The signal coming out of the RETIMING LOGIC is fed into 26LS30s and pulse transformers to drive the twisted pair lines (See section 5.0 for details).

6.1.9 HEADER/INTERMEDIATE HUB SWITCH

As seen on Figure 43 this hub can be configured as either an intermediate hub, or a Header one. One of the phone jacks, more specifically JACK #5 is either an input port or an output one. In order to implement this function, an 8 position DIP SWITCH (SW1) is used. The phone jacks are marked with UD, DD notation, meaning upstream data, and downstream data respectively. As specified in the StarLAN 1BASE5 draft NODES transmit data on UD pair, and HUBS on the DD pair. Switch SW1 has the function to invert UD, DD in PHONE JACK #5 to enable it to be either input or output port.

6.1.10 JABBER FUNCTION

This design does not implement the jabber unit but it is described here for completeness. IEEE 802.3 does not mandate this feature, but it is "Strongly Recommended". The jabber function in the HUB protects the network from abnormally long transmissions by any node.

Two timers T1, T2 are used by the JABBER function. They may be implemented either as local timers (one for each HUB port) or as global timers shared by all ports. After detecting an input active, timers T1, T2 will be started, and T1 will time out after 25 to 50 ms. T2 will time-out after 51 to 100 ms. During T2 time, after T1 expired, the HUB will send the CP-PATTERN informing any jamming stations to quit their transmissions. If on T2 time-out there are still jamming ports, their input is going to be disabled. A disabled port, will be reenabled whenever its input becomes again active and the downward side is idle.

The following is an explanation of the requirement that the downward side be idle to reenable an input port. Consider the case of Figure 54. The figure shows a two port HUB. Port A has two wires A_u, A_d for the up and down paths. Port B has B_u, B_d respectively. Port C is the output port, that broadcasts to the other HUBs higher in the hierarchy. Consider the case as shown, where B_u and B_d are shorted together. Suppose the case that port A_u is active. Its signal will propagate up in the hierarchy through C_u and come down from C_d to A_d, and B_d. Due to the short between B_d and B_u the signal will start a loop, that will first cause a collision and jam the network forever. This kind of fault is taken care of by the jabber circuitry. T1 and T2 will expire, causing the jabber logic to disable B_u input. Upon this disabling B_u is going to go Idle and be a candidate for future enabling. Suppose now that A_u is once again active. If the reenable condition would not require C_d to be IDLE, B_u would be reenabled causing the same loop to happen once again. Note that in this case C_d will be active before B_u causing this port to continue to be disabled and avoiding the jamming situation (Figure 55) gives a formal specification of the jabber function).

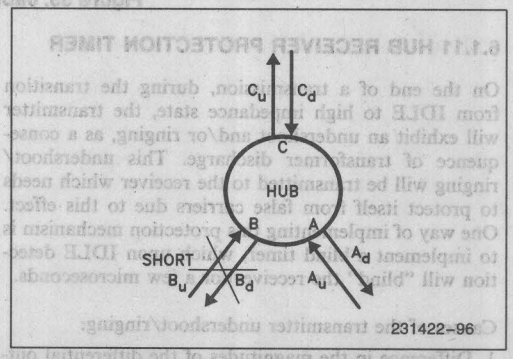


Figure 54. Jabber Function

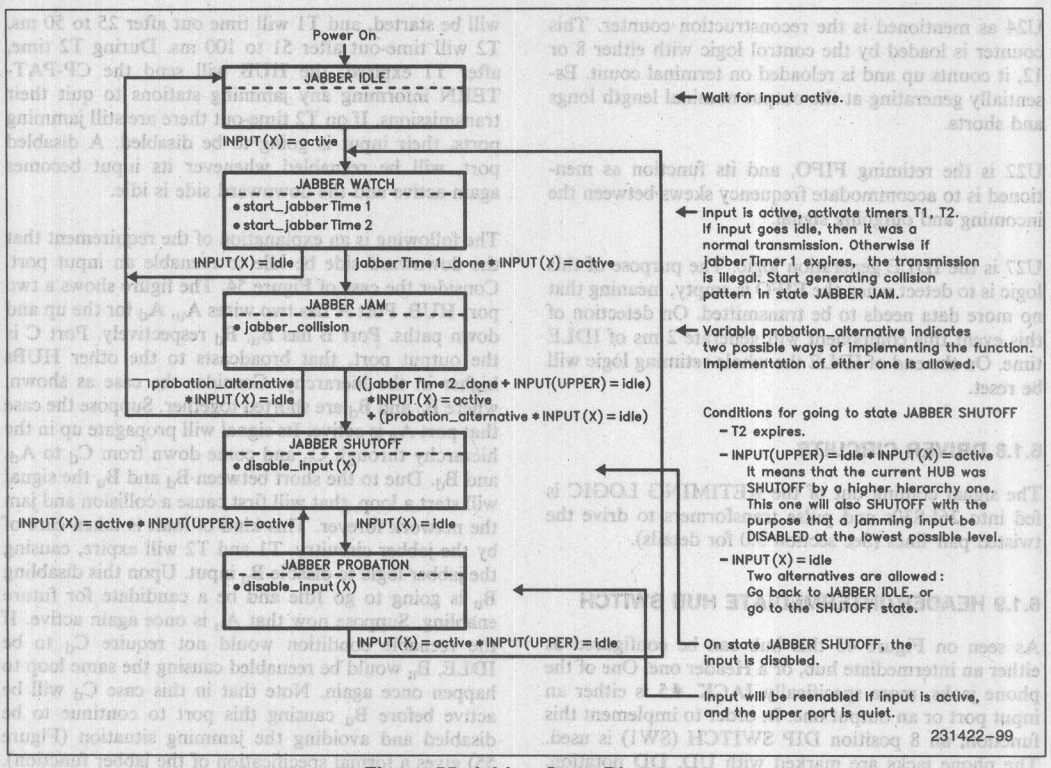


Figure 55. Jabber State Diagram

6.1.11 HUB RECEIVER PROTECTION TIMER

On the end of a transmission, during the transition from IDLE to high impedance state, the transmitter will exhibit an undershoot and/or ringing, as a consequence of transformer discharge. This undershoot/ringing will be transmitted to the receiver which needs to protect itself from false carriers due to this effect. One way of implementing this protection mechanism is to implement a blind timer, which upon IDLE detection will "blind" the receiver for a few microseconds.

Causes of the transmitter undershoot/ringing:

1. Difference in the magnitudes of the differential output voltage between the high and the low output stages.
2. Waveform assymmetry due to transmitter jitter.
3. Transmitter and receiver inductance (transformer L).
4. Two to three microseconds of IDLE pattern.

All the described elements will contribute to energy storage into the transformer inductor, which will discharge during the transition of the driver to high impedance.

The blinding timer is currently defined to be from 20 to 30 microseconds for the HUBs, being from 0 to 30 microseconds for the nodes (optional). The 82588 has built-in this function. It won't receive any frames for an inter-frame-spacing (IFS) from the idle detection.

6.1.12 HUB RELIABILITY

Since the StarLAN HUBs form focal points in the network, it is important for them to be very reliable, since they are single points of failure which can affect a number of nodes or can even bring down the whole network. StarLAN 1BASE5 draft requires HUBs to have a mean time between failures (MTBF) of at least 5 years of continuous operation.

7.0 SOFTWARE DRIVER

The software needed to drive the 82588 in a StarLAN environment is not different from that needed in a generic CSMA/CD environment. This section goes into specific procedures used for operations like TRANSMIT, RECEIVE, CONFIGURE, DUMP, ADDRESS SET-UP, etc. A special treatment will be given to interfacing with the IBM PC—DMA, interrupt and I/O.

Since all the routines were written and tried out in PLM-86 and ASM-86, all illustrations are in these languages.

The following software examples are pieces of an 82588 exerciser program. This program's main purpose was to exercise the 82588 functionality and provide the functions of traffic generation and monitoring. By such the emphasis was on speed and accuracy of statistics gathering.

7.1 Interfacing to IBM PC

The StarLAN board interfaces to the CPU, DMA controller and the interrupt controller on the IBM PC system board. The software to operate the 82588 runs on the system board CPU. The illustrated routines in this section show exactly how the software interface works between the system resources on the IBM PC and the StarLAN board.

7.1.1 DOING I/O ON IBM PC

The safest way to use the PC monitor as an output device and the keyboard as the input device is to use them through DOS system calls. The following is a set of routines which are handy to do most of the I/O:

key$stat	—to find out if a new key has been pressed
keyin$noecho	—to read a key from the keyboard
char$out	—to display a character on the screen
msg$out	—to display a character string on the screen
line$in	—to read in a character string from the keyboard

The exact semantics and the protocol for doing these functions through DOS system calls is shown in the listing in Figure 56. Refer to the DOS Manual for a more detailed description. To make a DOS system call, register AH of 8088 is loaded with the call Function Number and then, a software interrupt (or trap) 21 hex is executed. Other 8088 registers are used to transfer any parameters between DOS and the calling program. The code is written in Assembly language for register access. Let us see an example of the 'msg$out' routine:

```
lds dx,STRING_POINTER    ; load pointer to string in reg. ds:dx
mov ah,09h               ; 9 = function number for string o/p
int 21h                  ; DOS System Call
```

These procedures are called from another module, written in a higher level language like PLM-86. The parameters are transferred to the ASM-86 routines on the stack.

Examples of using the I/O routines:

```
KEY_STATUS = key$stat;                    /* INQUIRE KEYBOARD STATUS      */
NEW_KEY = keyin$noecho;                   /* INPUT NEW KEY                */
call line$in(@LINE_BUFFER);               /* STRING INPUT                 */
call char$out(CHAR_OUT);                  /* TO OUTPUT CHAR_OUT ON SCREEN*/
call msg$out(@('THIS IS A MESSAGE.$'));   /* OUTPUT STRING                */
                                          /* NOTE $ TERMINATOR            */
```

```
/*-----------------------------------------------------------------*/
/*     Declarations for external IBM PC I/O routines               */
/*-----------------------------------------------------------------*/

key$stat: procedure byte external;          /* key status routine */
end key$stat;

key$in$noecho: procedure byte external;     /* console input routine */
end key$in$noecho;

char$out: procedure(char) external;    /* console output routine */
declare char byte;
end char$out;

msg$out: procedure(msg$ptr) external;          /* console string output routine
declare msg$ptr pointer;
end msg$out;

line$in: procedure(line$ptr) external;         /* console string input routine
declare line$ptr pointer;
end line$in;
```

Assembly Language implementation of the routines

```
$TITLE(IBM/PC  DOS CALLS PROCEDURES)
        NAME    DOSPROCS
;
DGROUP  GROUP   DATA
CGROUP  GROUP   CODE
;
DATA    SEGMENT WORD PUBLIC 'DATA'
DATA    ENDS
;
DOS     EQU     21H
;
CODE    SEGMENT WORD PUBLIC 'CODE'
        ASSUME  CS:CGROUP,DS:DGROUP
```

231422-58

```
;
; CHAR$OUT: PROCEDURE(CHAR) EXTERNAL;
; DECLARE CHAR BYTE;
; END CHAR$OUT;
; Outputs character to the screen.
; DOS system call 2
;
CHAR            EQU     [BP+4]          STACK
;
CHAROUT         PROC    NEAR            +-------+
        PUBLIC  CHAROUT                 ! CHAR  ! x
        PUSH    BP                      +-------+
        MOV     BP,SP                   !IP lo  ! x-1
        MOV     DL,CHAR                 +-------+
        MOV     AH,2                    !IP hi  ! x-2
        INT     DOS                     +-------+
        POP     BP                      !BP lo  ! x-3
        RET     2                       +-------+
CHAROUT         ENDP                    !BP hi  ! x-4  <--SP
;                                       +-------+
; KEYIN$NOECHO: PROCEDURE BYTE EXTERNAL;
; END KEYIN$NOECHO;
; Reads character without echoing to display
;

KEYINNOECHO PROC    NEAR
        PUBLIC  KEYINNOECHO
        MOV     AH,8                    (DOS call 8)
        INT     DOS
        RET
KEYINNOECHO ENDP
```

Figure 7-56. I/O Routines for IBM/PC (continued)

231422-59

Figure 56. I/O Routines for IBM/PC

```
;
;
; MSG$OUT: PROCEDURE(MSG$PTR) EXTERNAL;
; DECLARE MSG$PTR POINTER;
; END MSG$OUT;
; /* NOTE: MESSAGE IS TERMINATED WITH A DOLLAR SIGN */
; MSG$PTR is double word pointer SEG:OFFSET

MSG_L           EQU     [BP+4]
MSG_H           EQU     [BP+6]
;
MSGOUT          PROC    NEAR
        PUBLIC  MSGOUT
;
        PUSH    BP
        MOV     BP,SP
        MOV     DX,MSG_L
        PUSH    DS
        MOV     AX,MSG_H
        MOV     DS,AX
        MOV     AH,9                    (DOS call 9)
        INT     DOS
        POP     DS
        POP     BP
        RET     4
MSGOUT          ENDP

;
; LINE$IN: PROCEDURE(LINE$PTR) EXTERNAL;
; DECLARE LINE$PTR POINTER;
; END LINE$IN
;

LINE_L          EQU     [BP+4]
LINE_H          EQU     [BP+6]
;
LINEIN          PROC    NEAR
        PUBLIC  LINEIN
        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        MOV     AX,LINE_H
        MOV     DS,AX
        MOV     DX,LINE_L
        MOV     AH,10                   (DOS call 10)
        INT     DOS
        POP     DS
        POP     BP
        RET     4
LINEIN          ENDP
                                                    231422-60

;
; KEY$STAT: PROCEDURE BYTE EXTERNAL;
; END KEY$STAT;
; Indicates whether any keyboard key was pressed.

KEYSTAT         PROC    NEAR
        PUBLIC  KEYSTAT
        MOV     AH,11                   (DOS call 11)
        INT     DOS
        RET
KEYSTAT         ENDP
;
;
CODE            ENDS
        END
                                                    231422-61
```

Figure 56. I/O Routines for IBM/PC (Continued)

7.2 Initialization and Declarations

Figure 57 shows some declarations describing what addresses the devices have and also some literals to help understand the other routines in this section.

Figure 58 shows the initialization routines for the IBM PC and for the 82588. It also shows some of the typical values taken by the memory buffers for Configure, IA_Set, Multicast and transmit buffers.

Following are some literal declarations that are used in the procedure examples

```
Following are some literal  declarations  that  are  used  in  the
procedure examples
  declare

   os_588          literally '0300h' , /* 82588 COMMAND/STATUS       */
   brd_port            literally '0301h' , /* DMA/INTERUPT ENABLE PORT   */
   pic_mask            literally '021h'  , /* 8259A MASK REGISTER        */
   pic_ocw2            literally '020h'  , /* 8259A COMMAND WORD 2       */
   dma_mask            literally '0ah'   , /* 8237A MASK REGISTER        */
   dma_mode            literally '0bh'   , /* 8237A MODE REGISTER        */
   dma_flff            literally '0ch'   , /* 8237A 1ST/2ND BYTE FLOP    */
   dma_addr_1          literally '02h'   , /* 8237A CHANNEL 1 ADDR. REG. */
   dma_bc_1            literally '03h'   , /* 8237A CHANNEL 1 BYTE COUNT */
   dma_addrh_1         literally '083h'  , /* CHANNEL 1 PAGE REGISTER    */
   dma_addr_3          literally '06h'   , /* 8237A CHANNEL 3 ADDR. REG. */
   dma_bc_3            literally '07h'   , /* 8237A CHANNEL 3 BYTE COUNT */
   dma_addrh_3         literally '082h'  , /* CHANNEL 3 PAGE REGISTER    */
   dma_on_1            literally '01h'   , /* START CHANNEL 1            */
   dma_on_3            literally '03h'   , /* START CHANNEL 3            */
   dma_off_1           literally '05h'   , /* STOP CHANNEL  1            */
   dma_off_3           literally '07h'   , /* STOP CHANNEL  3            */
   enable_588          literally '0dfh'  , /* UNMASK INTERRUPT LEVEL 5   */
   seoi_pico           literally '065h'  , /* SPECIFIC EOI LEVEL 5       */
   tx_dir              literally  '1'    , /* MEMORY TO 82588            */
   rx_dir              literally  '0'    , /* 82588 TO MEMORY            */
   dma_rx_mode_1       literally '045h'  , /* RX ON CHANNEL # 1          */
   dma_rx_mode_3       literally '047h'  , /* RX ON CHANNEL # 3          */
   dma_tx_mode_1       literally '049h'  , /* TX ON CHANNEL # 1          */
   dma_tx_mode_3       literally '04bh'  , /* TX ON CHANNEL # 3          */
                                                                  231422-62
```

Figure 57. Literal Declarations

Initialization Routines

```
Initialization routines

/* SYSTEM INITIALIZE */

sys_init: procedure;

   call set$interrupt (13,intr_588);                /* BASE 8, LEVEL 5    */
   output(pic_mask) = input(pic_mask) and enable_588; /* ENABLE 588 INTERR. */
   output(pic_ocw2) =    seoi_pico;                 /* ACKS PENDING INTERR*/

   wr_ptr,rd_ptr,fifoont=0;                         /* RESET STATUS FIFO  */

   /*************************************************/
   /* CONVERT SEG:OFFSET FORMAT TO 20 BIT ADDRESSES   */
   /* FOR ALL THE BUFFERS                             */
   /*************************************************/

   iaset_dma_addr = convert_20bit_addr(@ia_set_buff_588(0));
   cnf_dma_addr   = convert_20bit_addr(@config_588(0));
   dmp_dma_addr   = convert_20bit_addr(@dump_buff_588(0));
   mc_dma_addr    = convert_20bit_addr(@multicast_buff_588(0));
   tx_dma_addr    = convert_20bit_addr(@tx_buffer_588(0));
   do i=0 to 7 ;
     rx_dma_addr(i)=convert_20bit_addr(@rx_buffer(i).buff(0));
   end;

   output(brd_port)=0ffh;  /* ENABLE DMA AND INTERRUPT DRIVERS */

   end sys_init;

82588 initialization

init_588: procedure;

   config_588(00) = 10;          /* TO CONFIGURE ALL 10 PARAMETERS          */
   config_588(01) = 00;
   config_588(02  = 00001000b;   /* MODE 0, 8  MHZ CLOCK, 1 MB/S            */
   config_588(03) = buff_len/4;  /* RECEIVE BUFFER LENGTH                   */
   config_588(04) = 00100110b;   /* NO LOOPBACK, ADDR LEN = 6, PREAMBLE = 8 */
   config_588(05) = 00000000b;   /* DIFFERENTIAL MANCHESTER = OFF           */
   config_588(06) = 96;          /* IFS = 96 TCLK                           */
   config_588(07) = 0;           /* SLOT TIME = 512 TCLK                    */
   config_588(08) = 11110010b;   /* MAX. NO. RETRIES = 15                   */
   config_588(09) = 00000100b;   /* MANCHESTER ENCODING                     */
   config_588(10) = 10001000b;   /* INTERNAL CRS AND CDT, CRSF = 0          */
   config_588(11) = 64;          /* MIN FRAME LENGTH = 64 BYTES = 512 BITS  */
                                                                  231422-63
```

Figure 58. Initialization Routines

```
ia_set_buff_588(0) = 6;
ia_set_buff_588(1) = 0;
ia_set_buff_588(2) = 000h;
ia_set_buff_588(3) = 041h;
ia_set_buff_588(4) = 000h;
ia_set_buff_588(5) = 000h;
ia_set_uff_588(6)  = 000h;
ia_set_buff_588(7) = 000h;

multicast_buff_588(00) = 12;
multicast_buff_588(01) = 00h;
multicast_buff_588(02) = 11h;
multicast_buff_588(03) = 12h;
multicast_buff_588(04) = 13h;
multicast_buff_588(05) = 14h;
multicast_buff_588(06) = 15h;
multicast_buff_588(07) = 16h;
multicast_buff_588(08) = 21h;
multicast_buff_588(09) = 22h;
multicast_buff_588(10) = 23h;
multicast_buff_588(11) = 24h;
multicast_buff_588(12) = 25h;
multicast_buff_588(13) = 26h;

tx_buffer_588(00) = tx_frame_len mod 256;
tx_buffer_588(01) = tx_frame_len / 256;
tx_buffer_588(02) = 011h;   /* INITIAL DESTINATION ADDRESS - MC(1) */
tx_buffer_588(03) = 012h;
tx_buffer_588(04) = 013h;
tx_buffer_588(05) = 014h;
tx_buffer_588(06) = 015h;
tx_buffer_588(07) = 016h;

end init_588;
```

231422-64

Figure 58. Initialization Routines (Continued)

7.3 General Commands

Operations like Transmit, Receive, Configure, etc. are done by a simple sequence of loading the DMA controller with the necessary parameters and then writing the command to the 82588.

Example: Configure Command

To configure the operating environment of the 82588. This command must be the first one to be executed after a RESET.

```
call
DMA_LOAD(1,1,12,@CONFIG_588_ADDR);
output (CS_588) = 12h;
```

The first statement is the prologue to the configure command to the 82588 which calls a routine to load and initialize the DMA controller for the desired operation. This routine is described in section 7.4. The parameters for DMA_LOAD are:

```
first parameter  = 82588 channel
                   number ( = 1)
second parameter = direction ( = 1,
                   memory >> 82588)
third parameter  = length of DMA
                   transfer ( = 12)
fourth parameter = pointer to a 20 bit
                   address of the
                   memory buffer
                   (=@CONFIG_588_ADDR)
```

The second statement writes 12h to the command register of the 82588 to execute a Configure command on channel 1.

When the command execution is complete (successfully or not), 82588 interrupts the 8088 CPU through the 8259A, on the system board. This executes the interrupt service routine, described in section 7.5, which takes the epilogue action for the command.

Most operations are very similar in structure to Configure. The 82588 Reference Manual describes them in detail. Figure 59 shows a listing of the most commonly used operations like:

```
CONFIGURE     INDIVIDUAL-ADDRESS (IA)
              SET-UP
TRANSMIT      MULTICAST-ADDRESS (MC)
              SET-UP
DIAGNOSE      RECEIVE (RCV)-ENABLE
DUMP          RECEIVE (RCV)-DISABLE
TDR           RECEIVE (RCV)-STOP
RETRANSMIT    READ-STATUS
```

```
ia_set: procedure public;                    /* COMMAND - 01 */

    call dma_load(cmd_channel,tx_dir,8,@iaset_dma_addr);

    /* SET DMA CHANNEL 0 OR 1 TO TRANSFER FROM MEMORY
       TO THE 82588. iaset_dma_addr VARIABLE STORES THE
       20 BIT POINTER TO THE INDIVIDUAL ADDRESS BUFFER */

    if cmd_channel then output (cs_588) = 11h;
    else output(cs_588) = 01h;

    /* EVERY COMMAND CAN BE EXECUTED IN EITHER DMA CHANNEL 0 OR 1.
       THE VARIABLE cmd_channel INDICATES THE REQUIRED CHANNEL  */

end ia_set;

/*-------------------------------------------------------------------------*/

config: procedure public;                    /* COMMAND - 02 */

    call dma_load(cmd_channel,tx_dir,12,@cnf_dma_addr);
    if cmd_channel then output (cs_588) = 12h;
    else output(cs_588) = 02h;

end config;

/*-------------------------------------------------------------------------*/

multicast: procedure public;                 /* COMMAND - 03 */

    call dma_load(cmd_channel,tx_dir,14,@mc_dma_addr);
    if cmd_channel then output (cs_588) = 13h;
    else output(cs_588) = 03h;

end multicast;

/*-------------------------------------------------------------------------*/

transmit: procedure(buffer_len) public;      /* COMMAND - 04 */
    declare buffer_len word;

    tx_buffer_588(00) = low(buffer_len);
    tx_buffer_588(01) = high(buffer_len);

    call dma_load(cmd_channel,tx_dir,1536,@tx_dma_addr);

    if cmd_channel then output (cs_588) = 14h;
    else output(cs_588) = 04h;

end transmit;
```

231422-65

Figure 59. General Commands

```
end diagnose;

/*-------------------------------------------------------------------------*/

rcv_enable: procedure(channel,buffer_no,len) public;    /* COMMAND - 08 */

    declare channel byte;
    declare len  word;
    declare buffer_no byte;
    call dma_load(channel,rx_dir,len,@rx_dma_addr(buffer_no));
    if rx_channel then output (cs_588) = 18h;
    else output(cs_588) = 08h;

end rcv_enable;

/*-------------------------------------------------------------------------*/

rcv_disable: procedure public;              /* COMMAND - 10 */

    enable_rcv=0;
    output(cs_588)=0ah;

end rcv_disable;                                                  231422-66

/*-------------------------------------------------------------------------*/

rcv_stop: procedure public;                 /* COMMAND - 11 */

    enable_rcv=0;
    output(cs_588)= 0bh;

end rcv_stop;

/*-------------------------------------------------------------------------*/

retransmit: procedure public;               /* COMMAND - 12 */

    call dma_load(cmd_channel,tx_dir,1536,@tx_dma_addr);
    if cmd_channel then output (cs_588) = 1ch;
    else output(cs_588) = 0ch;

end retransmit;

/*-------------------------------------------------------------------------*/

abort: procedure public;                    /* COMMAND - 13 */

    output(cs_588)= 1dh;
    call new_status(1);

end abort;

/*-------------------------------------------------------------------------*/

reset_588: procedure public;                /* COMMAND - 14 */

    enable_rcv=0;
    output(cs_588) = 1eh;
    call config;

end reset_588;
                                                                  231422-67
```

Figure 59. General Commands (Continued)

7.4 DMA Routines

DMA_LOAD procedure is used to program the 8237A DMA controller for all the operations requiring DMA service. It also starts or enables the programmed DMA channel after programming it. Figure 60 shows the listing of this procedure. It accepts 4 parameters from the calling routine to decide the programming configuration for the 8237A. The parameters for DMA_LOAD are: Channel, direction, buff_len, and buff_addr.

```
Converting a pointer SEG:OFFSET to a 20 bit address
convert_20bit_addr:procedure(ptr) dword public;

   declare   ptr       pointer,
             ptr_addr  pointer,
             ptr_20bit dword   ,
             (wrd  based ptr_addr)(2) word;

   ptr_addr=@ptr;
   ptr_20bit=shl((ptr_20bit:=wrd(1)),4)+wrd(0);
   return(ptr_20bit);

end convert_20bit_addr;


IBM/PC DMA loading procedure

dma_load: procedure(channel,direction,buff_len,buff_addr) reentrant public;

    declare channel byte;              /* CHANNEL #, 0 or 1                */
    declare direction byte;            /* 0-RX, 588 -> MEM; 1-TX, MEM -> 588 */
    declare buff_len word;             /* BYTE COUNT                       */
    declare buff_addr pointer;         /* BUFFER ADDR IN 20 BITS FORM      */
    declare (wrd based buff_addr)(2) word;

    channel=channel and 1;             /* GET LEAST SIGNIFICANT BIT        */

    if channel=0 then                  /* EXECUTE COMMAND ON CHANNEL 1     */
      do;
      output(dma_flff)  = 0;           /* CLEAR FIRST/LAST FLIP-FLOP       */
      if direction=0
          then output(dma_mode)=dma_rx_mode_1;  /* DIRECTION BIT, TELLS    */
      else output(dma_mode)=dma_tx_mode_1;      /* TRANSMIT OR RECEIVE     */
      output(dma_addr_1)  = low (wrd(0));       /* LOAD LSB ADDRESS BYTE   */
      output(dma_addr_1)  = high(wrd(0));       /* LOAD MSB ADDRESS BYTE   */
      output(dma_addrh_1) = low (wrd(1));       /* LOAD PAGE REGISTER      */
      output(dma_bo_1)    = low (buff_len);     /* LOAD LSB BYTE COUNT     */
      output(dma_bo_1)    = high(buff_len);     /* LOAD MSB BYTE COUNT     */
      output(dma_mask) = dma_on_1;              /* START CHANNEL 1         */
      end;
    else do;                          /* SAME AS BEFORE FOR CHANNEL 3     */
      output(dma_flff)  = 0;
      if direction=0
          then output(dma_mode)=dma_rx_mode_3;
      else output(dma_mode)=dma_tx_mode_3;
      output(dma_addr_3)  = low (wrd(0));
      output(dma_addr_3)  = high(wrd(0));
      output(dma_addrh_3) = low (wrd(1));
      output(dma_bo_3)    = low (buff_len);
      output(dma_bo_3)    = high(buff_len);
      output(dma_mask) = dma_on_3;
      end;

end dma_load;
```

231422-68

Figure 60. DMA Routine

One peculiarity about this procedure is that in order to speed up the DMA step-up, this procedure doesn't get a pointer to the buffer, but a pointer to a 20 bit address in the 8237 format. The 8088/8086 architecture define pointers as 32 bits seg:offset entities, where seg and offset are 16 bit operands. By the other hand the IBM/PC uses an 8237A and a page register, requiring a memory address to be a 20 bit entity. The process of converting a seg:offset pointer to a 20 bit address is time consuming and could negatively affect the performance of the 82588 driver software. The decision was to make the pointer/address conversions during initialization, considering that the buffers are static in memory (essentially removing this calculation from the real time response loops).

Figure 61 is a listing of the DMA_LOAD procedure for the 80188 or 80188 on-chip DMA controller. It has the same caller interface as the 8237A based one.

```
dma_load: procedure(channel,direction,trans_len,buff_addr) reentrant;

/* To load and start the  80186 DMA controller for the desired operation */

  declare dma_rx_mode   literally '1010001001000000b'; /* rx channel */
        /* src=IO, dest=M(inc), sync=src, TC, noint, priority, byte */

  declare dma_tx_mode   literally '0000110100000000b'; /* tx channel */
        /* src=M(inc), dest=IO, sync=dest, TC, noint, noprior, byte */


  declare channel byte;       /* channel #                              */
  declare direction byte;     /* 0 = rx, 588 -> mem; 1 = tx, mem -> 588 */
  declare trans_len word;     /* byte count                             */
  declare buff_addr pointer; /* buffer pointer in 20 bit addr. form     */

  declare (wrd based buff_addr)(2) word;

  do case channel and 00000001b;
    do case direction and 00000001b;
      do;                /* channel 0, 588 to memory  */
      output(dma_0_dpl)  = wrd(0);
      output(dma_0_dph)  = wrd(1);
      output(dma_0_spl)  = ch_a_588;
      output(dma_0_sph)  = 0;
      output(dma_0_tc)   = trans_len;
      output(dma_0_cw)   = dma_rx_mode or 0006h; /* Start DMA chl 0 */
      end;

      do;               /* channel 0, memory to 588 */
      output(dma_0_dpl)  = ch_a_588;
      output(dma_0_dph)  = 0;
      output(dma_0_spl)  = wrd(0);
      output(dma_0_sph)  = wrd(1);
      output(dma_0_tc)   = trans_len;
      output(dma_0_cw)   = dma_tx_mode or 0006h; /* Start DMA chl 0 */
      end;
    end;
```

231422-69

Figure 61. 80186 DMA Routines

```
        do case direction and 00000001b;
            do;                     /* channel 1, 588 to memory */
            output(dma_1_dpl)  = wrd(0);
            output(dma_1_dph)  = wrd(1);
            output(dma_1_spl)  = ch_b_588;
            output(dma_1_sph)  = 0;
            output(dma_1_tc)   = trans_len;
            output(dma_1_cw)   = dma_rx_mode or 0006h; /* Start DMA ch1 1 */
            end;

            do;                     /* channel 1, memory to 588 */
            output(dma_1_dpl)  = ch_b_588;
            output(dma_1_dph)  = 0;
            output(dma_1_spl)  = wrd(0);
            output(dma_1_sph)  = wrd(1);
            output(dma_1_tc)   = trans_len;
            output(dma_1_cw)   = dma_tx_mode or 0006h; /* Start DMA ch1 1 */
            end;
        end;
    end;

end dma_load;
```

231422-70

Figure 61. 80186 DMA Routines (Continued)

7.5 Interrupt Routine

The interrupt service routine, 'intr_588', shown in Figure 62, is invoked whenever the 82588 interrupts. The main difficulty in designing this interrupt routine was to speed its performance. Fast status processing was a basic requirement to be able to handle back to back frames.

The interrupt handler will read 82588 status, and put them into a 64 byte long EVENT_FIFO. Those statuses are going to be used in the main loop for updating screen counters. All the statistics are updated as fast as possible in the interrupt handler to fulfill the back-to-back frame processing requirement.

The interrupt handler is not reentrant, interrupts are disabled at the beginning and reenabled on exit.

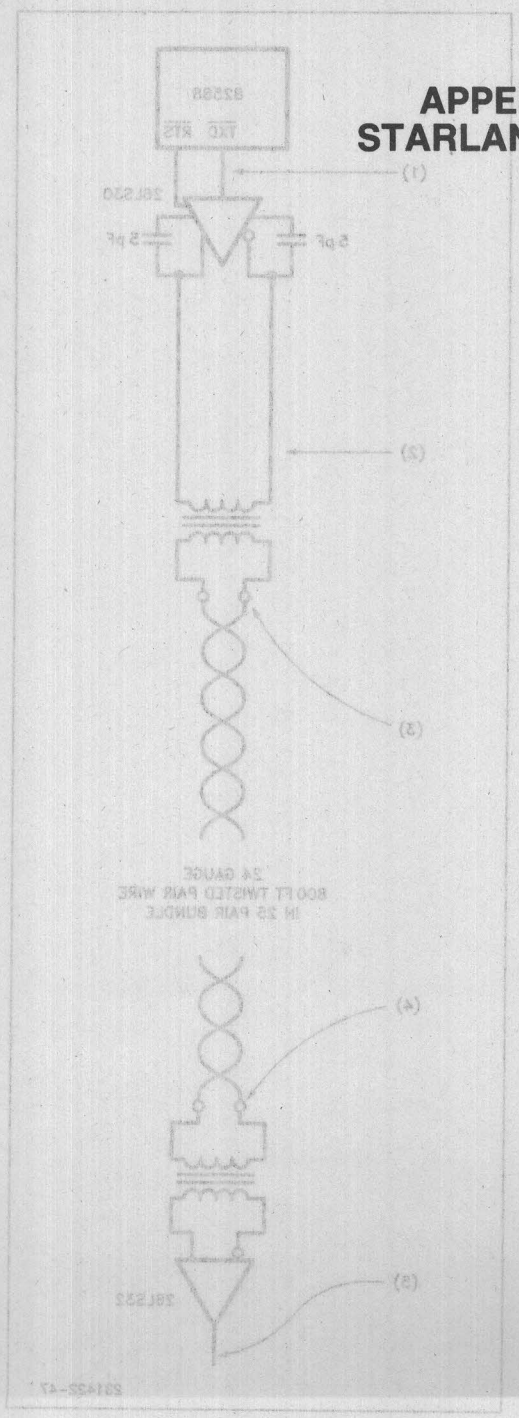
```
Interrupt service routine

intr_588:procedure interrupt 13;

    declare stat                    byte,
            event                   byte,
            i                       byte,
            (st0,st1,st2,st3)       byte,
            rx_st0                  byte,
            rx_st1                  byte;

   /* FOLLOWING LITERALS HAVE THE PURPOSE OF ENABLE ACTING
      ON EITHER CHANNEL 1 OR 3 SELECTIVELY                    */

   declare

     stop_cmd_dma  literally 'if cmd_channel
                                then output(dma_mask)=dma_off_3;
                              else output(dma_mask)=dma_off_1',
     stop_rx_dma   literally 'if rx_channel
                                then output(dma_mask)=dma_off_3;
                              else output(dma_mask)=dma_off_1',

     issue_rtx_cmd literally 'if cmd_channel
                                then output(os_588)=1Ch;
                              else output(os_588)=0ch',
     issue_tx_cmd  literally 'if cmd_channel
                                then output(os_588)=14h;
                              else output(os_588)=04h';

     disable;                                    /* DISABLE INTERRUPTS    */
                                                 /* NO INTERR. NESTING    */
     output(os_588)  =0fh;                       /* RLS 588 PTR, START 0 */

     event_fifo(wr_ptr).st0,st0=input(os_588); /* READ 82588 STATUS     */
     event_fifo(wr_ptr).st1,st1=input(os_588); /* REGISTERS, PASSING    */
     event_fifo(wr_ptr).st2,st2=input(os_588); /* THEM TO THE MAIN      */
     event_fifo(wr_ptr).st3,st3=input(os_588); /* PROGRAM ON THE FIFO   */

     wr_ptr=(wr_ptr+1) and 0fh;                  /* INCREMENT FIFO        */
     fifocnt=(fifocnt+1) and 0fh;                /* COUNTERS              */

     event=st0 and 0fh;                          /* GET EVENT FIELD       */

     output(os_588)=80h;                         /* ACKNOWLEDGE 82588     */
                                                 /* INTERRUPT             */   231422-71


   do case event;

   ev_00: ;                             /* NOP COMMAND          */
   ev_01: stop_cmd_dma;                 /* IA_SETUP, STOP DMA   */
   ev_02: stop_cmd_dma;                 /* CONFIGURE, STOP DMA  */
   ev_03: stop_cmd_dma;                 /* MULTICAST, STOP DMA  */
   ev_04: do;                           /* TRANSMIT DONE        */
          stop_cmd_dma;

          /* CHECK IF THERE WAS A COLLISION AND IS NOT THE
             MAX COLLISION                                     */

          stat=(st2 and 10000000b) or (st1 and 00100000b);
          if (stat=80h)
             then do;                   /* RETRANSMIT           */
                  call dma_load(cmd_channel,tx_dir,1536,@tx_dma_addr);
                  issue_rtx_cmd;
                  /* UPDATE STATISTICS                         */
                  total_tx_count=total_tx_count+1;
                  coll_cnt(17) = coll_cnt(17) + 1; /*TOTAL COLL*/
                  bad_tx_count = bad_tx_count + 1;
                  end;
          else do;
              if in_loop    /* EXECUTING TRANSMISSIONS IN LOOP  */
               then do;     /* RE ISSUE TRANSMIT COMMAND        */
                   call dma_load(cmd_channel,tx_dir,1536,@tx_dma_addr
                   issue_tx_cmd;
                   total_tx_count=total_tx_count+1;
                   end;
              if (st2 and 00100000b) = 0          /* BAD TRANSMIT*/
                then do;
                   bad_tx_count = bad_tx_count + 1;
                   /* INCREMENT UNDERRUN COUNTER               */
                   tmp=scr(tmp:=st2,1);
                   tx_under=tx_under plus 0;
                   /* INCREMENT LOST CTS COUNTER               */
                   tmp=scr(tmp,1);
                   lost_cts=lost_cts plus 0;
                   /* INCREMENT LOST CRS COUNTER               */
                   tmp=scr(tmp,1);
                   lost_crs=lost_crs plus 0;
                   if (stat=0A0h)  /* INC COLLISIONS COUNTER  */
                      then coll_cnt(17) = coll_cnt(17) + 1;
                   end;
              end;

              /* INCREMENT DEFER  COUNTER                      */
              tmp=scl((tmp:=st1),1);
              tx_defer=tx_defer plus 0;
          end;                                                 231422-72
```

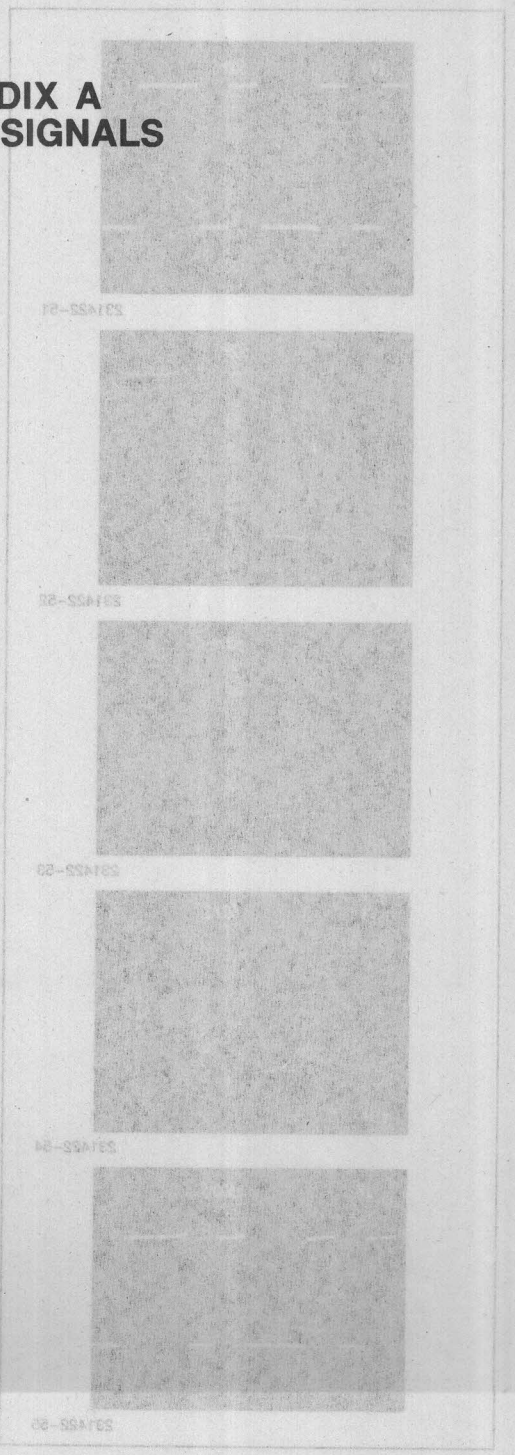
Figure 62. Interrupt Routine

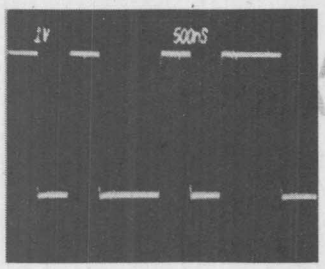
```
ev_05: stop_cmd_dma;                    /* TDR COMMAND, STOP DMA    */
ev_06: stop_cmd_dma;                    /* DUMP COMMAND, STOP DMA   */
ev_07: stop_cmd_dma;                    /* DIAGNOSE CMD, STOP DMA   */
ev_08:                                  /* RECEIVED FRAME           */
     do;
     stop_rx_dma;
     i=(current_buff+1) and 00000111b; /* INC BUFFER NO. MOD 8*/
     if enable_rov<>0                   /* IF RECEIVER IS ON        */
         then do;                       /* PREPARE NEXT BUFFER */
              call dma_load(rx_channel,rx_dir,1532,@rx_dma_addr(i));
              if rx_channel then output(os_588)= 18h;
              else output(os_588)=08h;
              rx_buffer(i).chain_cnt=0;
              end;
     else call rov_disable;             /* DISABLE RECEIVER     */

     /* FIND ADDRESS OF END OF CURRENTLY RECEIVED BUFFER       */
     /* BY CALCULATING IT WITH THE 82588 BYTE COUNT REGS.      */
     rx_buff_off=(shl(double(st2),8) or double(st1));
     /* READ STATUS BYTES FROM MEMORY                          */
     rx_st0=rx_buffer(current_buff).buff(rx_buff_off-2);
     rx_st1=rx_buffer(current_buff).buff(rx_buff_off-1);
     /* UPDATE ACTUAL BUFFER SIZE                              */
     rx_buffer(current_buff).actual_size=rx_buff_off;
     rx_buffer(current_buff).st0=rx_st0;
     rx_buffer(current_buff).st1=rx_st1;
     current_buff=i;
     /* UPDATE TOTAL RECEIVED BUFFERS                          */
     total_rcv_count=total_rcv_count+1;
     /* UPDATE STATISTICS                                      */
     if (rx_st1 and 00100000b)=0
         then do;
              bad_rcv_count=bad_rcv_count+1;
              /* INCREMENT  NO END OF FRAME COUNTER            */
              tmp=sor(tmp:=rx_st0,7);
              no_eof=no_eof plus 0;
              /* INCREMENT  SHORT FRAME COUNTER                */
              tmp=sor(tmp,1);
              srt_frm=srt_frm plus 0;
              /* INCREMENT  RX OVERRUN COUNTER                 */
              tmp=sor(tmp:=rx_st1,1);
              rx_over=rx_over plus 0;
              /* INCREMENT  ALIGNMENT ERROR COUNTER            */
              tmp=sor(tmp,2);
              alg_err=alg_err plus 0;
              /* INCREMENT  CRC ERROR COUNTER                  */
              tmp=sor(tmp,1);
              crc_err=crc_err plus 0;
              end;
     end;                                                      231422-73

      /*  EV_09 REQUESTS ASSIGNMENT OF A NEW BUFFER           */
     ev_09:  call allocate_new_buffer(not(rol(st3,1)) and 00000001b);
     ev_10:  stop_rx_dma;     /* RECEIVE DISABLE               */
     ev_11:  stop_rx_dma;     /* STOP RECEIVE                  */
     ev_12:  do;              /* RE-TRANSMIT DONE              */
             stat=(st2 and 10000000b) or (st1 and 00100000b);
             if (stat=80h)
             then do;         /* RETRANSMIT                    */
                  call dma_load(1,tx_dir,1536,@tx_dma_addr);
                  issue_rtx_cmd;
                  coll_cnt(17) = coll_cnt(17) + 1;
                  total_tx_count=total_tx_count+1;
                  bad_tx_count=bad_tx_count +1;
                  end;
             else do;
                  if in_loop
                   then do;   /* LOOP RETRANSMISSIONS          */
                        call dma_load(cmd_channel,tx_dir,1536,@tx_dma_a
                        issue_tx_cmd;
                        total_tx_count=total_tx_count+1;
                        end;
                  if (stat=0A0h) /* MAX COLLISION             */
                     then do;
                          coll_cnt(16) = coll_cnt(16)+1;
                          coll_cnt(17) = coll_cnt(17)+1;
                          bad_tx_count=bad_tx_count +1;
                          end;
                  /* UPDATE SPECIFIC COLLISION COUNTER         */
                  else  coll_cnt(st1 and 0fh)
                             = coll_cnt(st1 and 0fh) + 1;
                  end;
             end;
     ev_13:  stop_cmd_dma;      /* EXECUTION ABORTED           */
     ev_14:  ;
     ev_15: stop_cmd_dma;       /* DIAGNOSE FAILED             */
     end;

/*  ACKNOWLEDGE 8259A  INTERRUPT                               */
output(pic_ocw2)= seoi_pic;   /* SPECIFIC EOI FOR 8259        */

end intr_588;                                                  231422-74
```

Figure 62. Interrupt Routine (Continued)

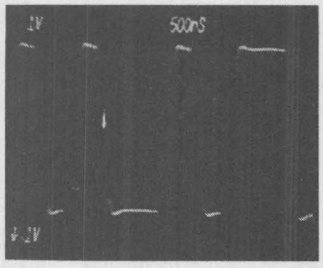
APPENDIX A
STARLAN SIGNALS



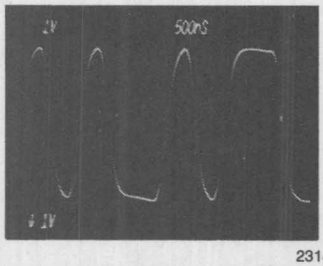




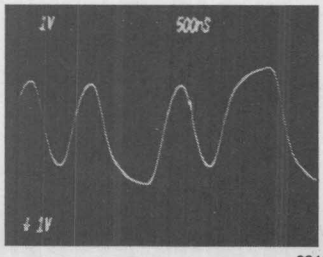
231422-51



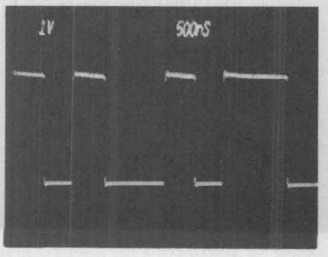
231422-52



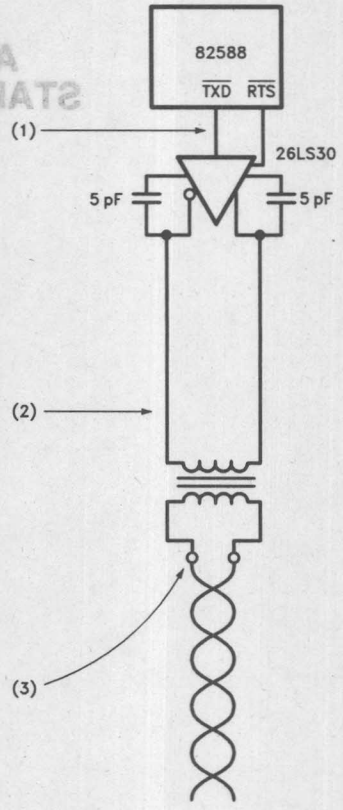
231422-53

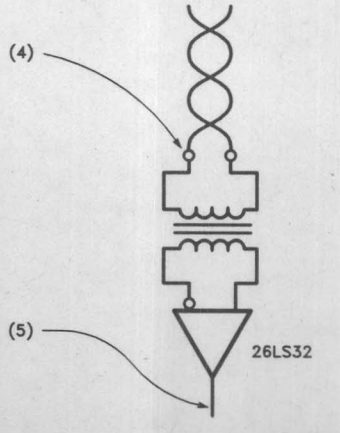


231422-54



231422-55





231422-47

Figure 63. StarLAN Signals

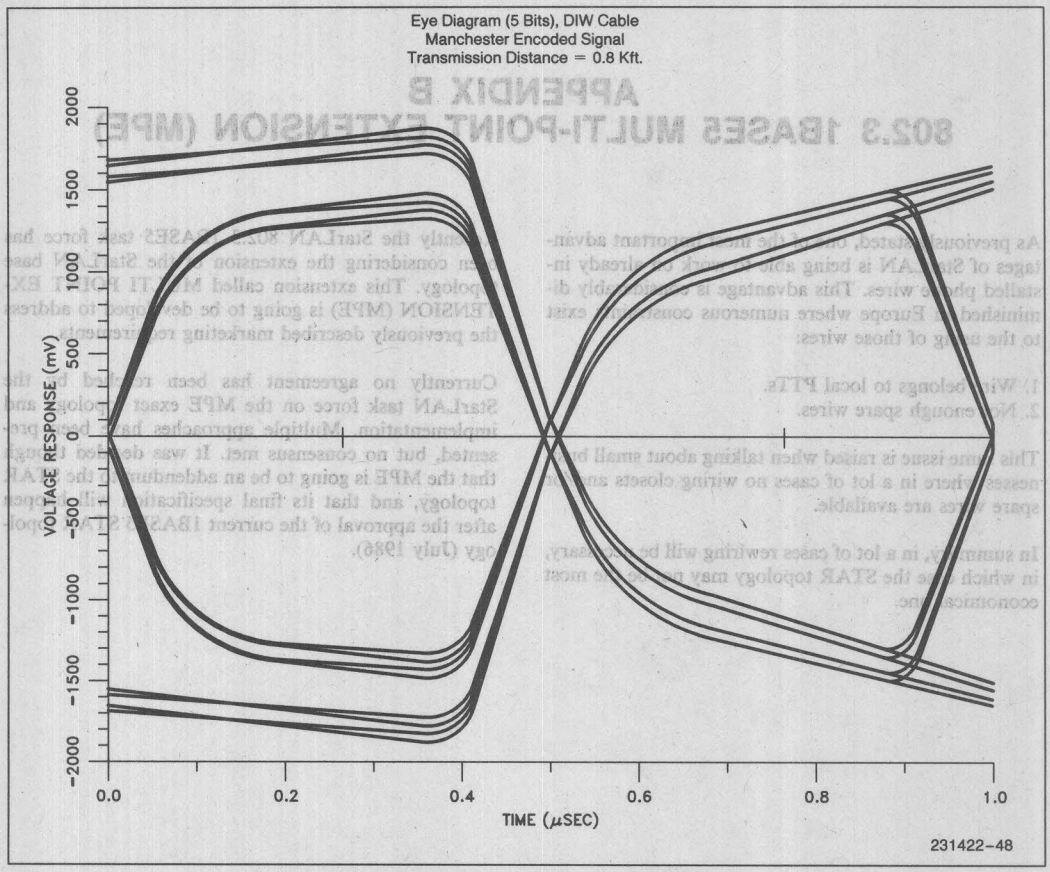


Figure 64. Received Signal Eye Diagram

APPENDIX B
802.3 1BASE5 MULTI-POINT EXTENSION (MPE)

As previously stated, one of the most important advantages of StarLAN is being able to work on already installed phone wires. This advantage is considerably diminished in Europe where numerous constraints exist to the using of those wires:

1. Wire belongs to local PTTs.
2. Not enough spare wires.

This same issue is raised when talking about small businesses where in a lot of cases no wiring closets and/or spare wires are available.

In summary, in a lot of cases rewiring will be necessary, in which case the STAR topology may not be the most economical one.

Recently the StarLAN 802.3 1BASE5 task force has been considering the extension of the StarLAN base topology. This extension called MULTI POINT EXTENSION (MPE) is going to be developed to address the previously described marketing requirements.

Currently no agreement has been reached by the StarLAN task force on the MPE exact topology and implementation. Multiple approaches have been presented, but no consensus met. It was decided though that the MPE is going to be an addendum to the STAR topology, and that its final specification will happen after the approval of the current 1BASE5 STAR topology (July 1986).

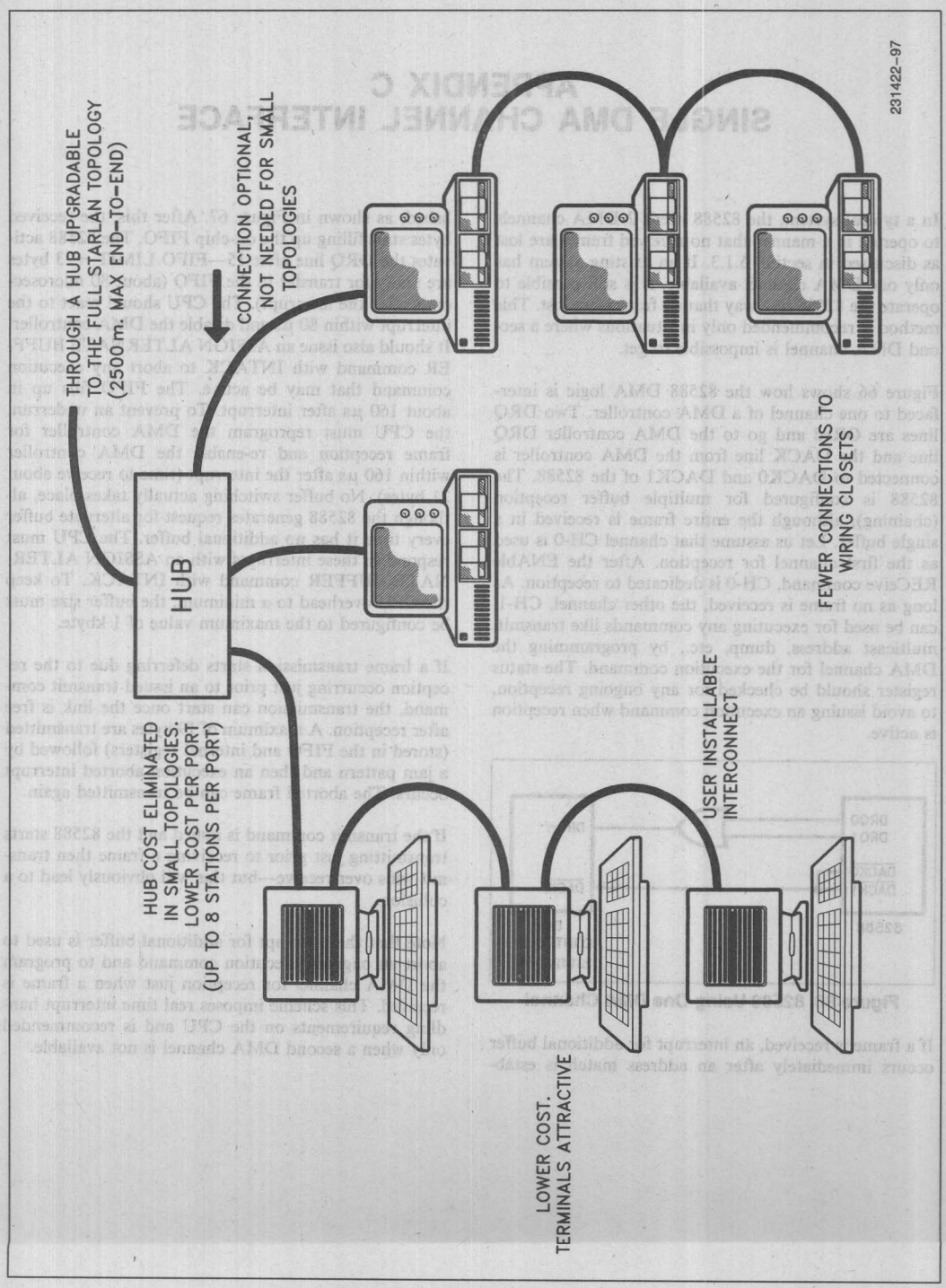


Figure 65. Multipoint Extension

APPENDIX C
SINGLE DMA CHANNEL INTERFACE

In a typical system, the 82588 needs 2 DMA channels to operate in a manner that no received frames are lost as discussed in section 5.1.3. If an existing system has only one DMA channel available, it is still possible to operate the 82588 in a way that no frames are lost. This method is recommended only in situations where a second DMA channel is impossible to get.

Figure 66 shows how the 82588 DMA logic is interfaced to one channel of a DMA controller. Two DRQ lines are ORed and go to the DMA controller DRQ line and the DACK line from the DMA controller is connected to DACK0 and DACK1 of the 82588. The 82588 is configured for multiple buffer reception (chaining), although the entire frame is received in a single buffer. Let us assume that channel CH-0 is used as the first channel for reception. After the ENAble RECeive command, CH-0 is dedicated to reception. As long as no frame is received, the other channel, CH-1, can be used for executing any commands like transmit, multicast address, dump, etc., by programming the DMA channel for the execution command. The status register should be checked for any ongoing reception, to avoid issuing an execution command when reception is active.

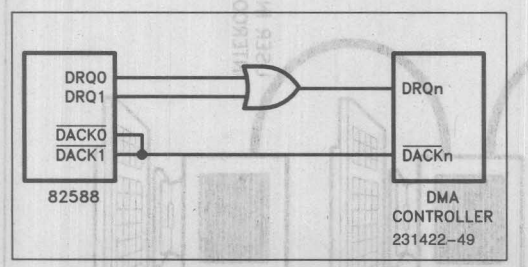


Figure 66. 82588 Using One DMA Channel

If a frame is received, an interrupt for additional buffer occurs immediately after an address match is established, as shown in Figure 67. After this, the received bytes start filling up the on-chip FIFO. The 82588 activates the DRQ line after 15—FIFO LIMIT + 3 bytes are ready for transfer in the FIFO (about 80 microseconds after the interrupt). The CPU should react to the interrupt within 80 μs and disable the DMA controller. It should also issue an ASSIGN ALTERNATE BUFFER command with INTACK to abort any execution command that may be active. The FIFO fills up in about 160 μs after interrupt. To prevent an underrun, the CPU must reprogram the DMA controller for frame reception and re-enable the DMA controller within 160 μs after the interrupt (time to receive about 21 bytes). No buffer switching actually takes place, although the 82588 generates request for alternate buffer every time it has no additional buffer. The CPU must respond to these interrupts with an ASSIGN ALTERNATE BUFFER command with INTACK. To keep the CPU overhead to a minimum, the buffer size must be configured to the maximum value of 1 kbyte.

If a frame transmission starts deferring due to the reception occurring just prior to an issued transmit command, the transmission can start once the link is free after reception. A maximum of 19 bytes are transmitted (stored in the FIFO and internal registers) followed by a jam pattern and then an execution aborted interrupt occurs. The aborted frame can be transmitted again.

If the transmit command is issued and the 82588 starts transmitting just prior to receiving a frame then transmit wins over receive—but this will obviously lead to a collision.

Note that the interrupt for additional buffer is used to abort an ongoing execution command and to program the DMA channel for reception just when a frame is received. This scheme imposes real time interrupt handling requirements on the CPU and is recommended only when a second DMA channel is not available.

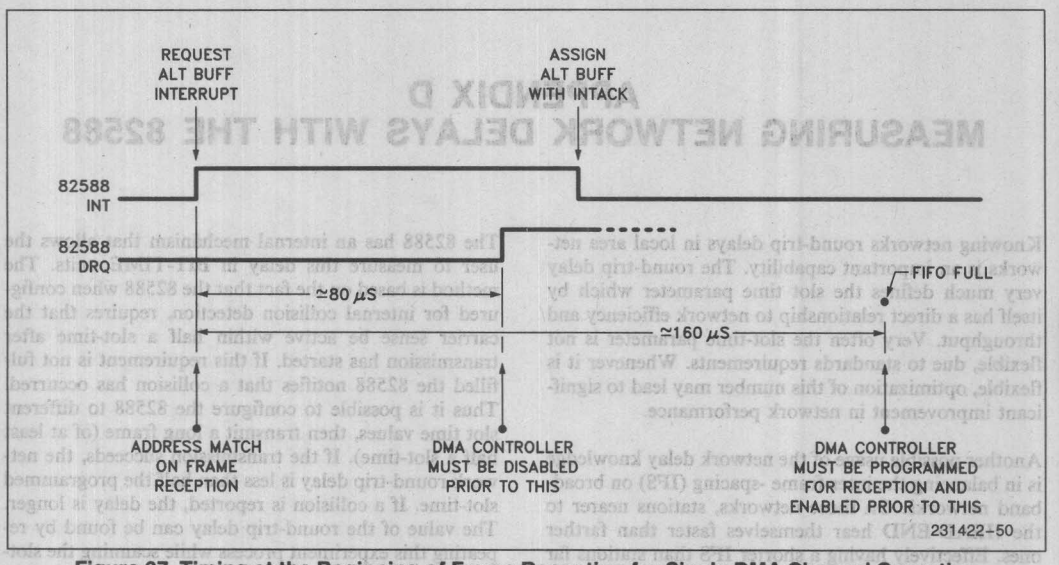


Figure 67. Timing at the Beginning of Frame Reception for Single DMA Channel Operation

APPENDIX D
MEASURING NETWORK DELAYS WITH THE 82588

Knowing networks round-trip delays in local area networks is an important capability. The round-trip delay very much defines the slot time parameter which by itself has a direct relationship to network efficiency and throughput. Very often the slot-time parameter is not flexible, due to standards requirements. Whenever it is flexible, optimization of this number may lead to significant improvement in network performance.

Another possible usage of the network delay knowledge is in balancing the inter-frame -spacing (IFS) on broadband networks. On those networks, stations nearer to the HEAD-END hear themselves faster than farther ones. Effectively having a shorter IFS than stations far from the HEAD-END. This difference causes an inbalance in network access time for different stations at different distances from the HEAD-END. Knowing the STATION/HEAD-END delay allows the user to reprogram the 82588 IFS accordingly, and by that balance the effective IFS for all the stations.

The 82588 has an internal mechanism that allows the user to measure this delay in BIT-TIME units. The method is based on the fact that the 82588 when configured for internal collision detection, requires that the carrier sense be active within half a slot-time after transmission has started. If this requirement is not fulfilled the 82588 notifies that a collision has occurred. Thus it is possible to configure the 82588 to different slot time values, then transmit a long frame (of at least half a slot-time). If the transmission succeeds, the network round-trip delay is less than half the programmed slot-time. If a collision is reported, the delay is longer. The value of the round-trip delay can be found by repeating this experiment process while scanning the slot-time configuration parameter value and searching the threshold. A binary search algorithm is used for that purpose. First the slot-time is configured for the maximum (2048 bits) and according if there was a collision or not, the number changed for the next try. (See Figure 68)

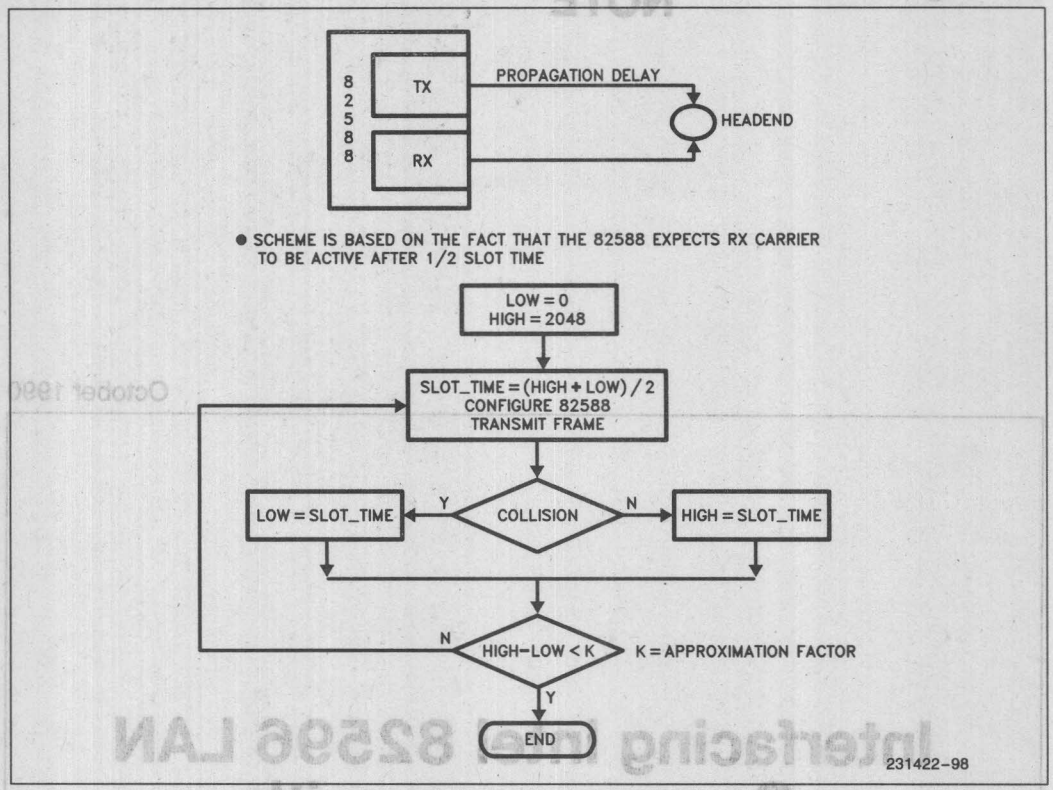
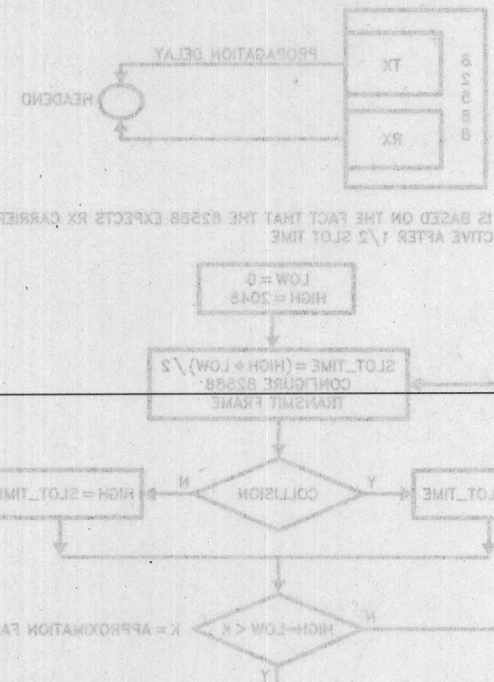


Figure 68. Network Delay Measurement using the 82588



October 1990

Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

Some portions of this document were provided by Dr. Design of San Diego, CA

Order Number: 292076-001

Interfacing Intel 82596 LAN Coprocessors with M68000 Family Microprocessors

CONTENTS

	PAGE
1.0 INTRODUCTION	1-497
1.1 Scope	1-497
1.2 Fundamental Assumptions	1-497
2.0 GENERIC IMPLEMENTATION ISSUES	1-497
2.1 Block Definitions	1-497
2.2 Clocking	1-499
2.3 Reset Retiming	1-499
2.4 CA and PORT Generation	1-500
2.5 Arbitration	1-501
2.5.1 Refresh Requests	1-501
2.5.2 82596 Requests	1-501
2.5.3 Arbiter Implementation	1-502
2.6 Signal Conversion	1-504
2.7 Wait State and Burst Generator	1-505
2.7.1 General Information	1-505
2.7.2 Single Cycle Bus Transfers	1-505
2.7.3 Burst-Cycle and Multiple-Cycle Bus Transfers	1-505
3.0 MC68030/82596CA INTERFACE	1-506
3.1 Design Specifications	1-506
3.2 Clocking	1-506
3.3 Reset Retiming	1-506
3.4 CA and PORT Generator	1-506
3.5 Bus Arbiter	1-507
3.6 Memory Signal Conversion	1-507
3.7 Wait State and Burst Generator	1-508
3.7.1 General Information	1-508
3.7.2 Single Cycle Transfers	1-508
3.7.3 Burst Cycle Bus Transfers	1-508
4.0 M68000/82596DX INTERFACE	1-509
4.1 Design Specifications	1-509
4.2 Clocking	1-509
4.3 Reset Timing	1-509
4.4 CA and PORT Generator	1-510
4.5 Bus Arbiter	1-510
4.6 Memory Signal Conversion	1-510
4.7 Wait State Generator	1-511
5.0 MC68000/82596SX INTERFACE	1-511
5.1 Design Specifications	1-511
5.2 Clocking	1-512
5.3 Reset Retiming	1-512
5.4 CA and PORT Generator	1-512
5.5 Bus Arbiter	1-512
5.6 Memory Signal Conversion	1-512
5.7 Wait State Generator	1-513
APPENDIX A SCHEMATICS	1-514
A.1 MC68030/82596CA	1-515
A.2 MC68020/82596DX	1-516
A.3 MC68000/82596SX	1-517
APPENDIX B PLD EQUATIONS	1-518
B.1 MC68030/82596CA	1-519
B.2 MC68020/82596DX	1-530
B.3 MC68000/82596SX	1-539
APPENDIX C TIMING DIAGRAMS	1-545
C.1 MC68030/82596CA	1-546
C.2 MC68020/82596DX	1-555
C.3 MC68000/82596SX	1-562
APPENDIX D PARTS LIST	1-569

Throughout this document, M68000 is used as a general reference to a family of microprocessors, which includes the MC68000, MC68020, MC68030. A reference to a particular member of the family will use the MC prefix followed by the specific number. 82596 is used as a general reference to a family of LAN coprocessors—the *82596CA, 82596DX, and 82596SX*. A reference to a particular member of the family will use 82596 followed by the two letter suffix.

Table of Figures and Tables

CONTENTS

PAGE

FIGURES

Figure 1 M68000 Family and 82596 Family Interface Generic Block Diagram 1-498
Figure 2 Clock Timing Relationships 1-498
Figure 3 Reset Retiming Block Diagram 1-499
Figure 4 CA and PORT Block Diagram 1-500
Figure 5 CA and PORT State Transition Diagram 1-500
Figure 6 Arbiter State Transition Diagram 1-503
Figure 7 Arbiter Signal Timings 1-503
Figure 8 Memory Signal Conversion Block Diagram 1-504
Figure 9 Memory Cycles for M68000 Family and 82596 Family 1-504
Figure 10 Wait-State and Burst Generation Block Diagram 1-505
Figure 11 82596DX and 82596SX Reset Retiming Circuit 1-509

TABLES

Table 1 82596-Family Parallel Bus Comparison 1-497
Table 2 Clocking Specifications 1-499
Table 3 Reset Specifications 1-500
Table 4 Arbitration Signal Input Timings 1-501
Table 5 Arbitration Signal Output Timings 1-502
Table 6 82596CA Worst-Case Reset Timing Margin 1-506
Table 7 82596DX/SX Worst-Case Reset Timing Margins 1-509

1.0 INTRODUCTION

1.1 Scope

The 82596 family of LAN coprocessors provide IEEE 802.3 MAC functions for use with 10BASE5 (Ethernet), 10BASE2 (Cheapernet), 10BASE-T (Twisted Pair Ethernet), 1BASE5 (StarLAN), and other CSMA/CD LANs with serial bit rates up to 20 Mb/s. The three members of the 82596 family differ only in the characteristics of their parallel interfaces; the FIFO and serial functions are identical. Table 1 shows the parallel bus differences.

This document describes the circuits required to interface the Intel 82596 family of LAN coprocessors with the M68000 family of microprocessors. First, general interface issues are identified and then three specific designs are provided—including the PLD equations, timing diagrams, and schematics.

- 82596CA and MC68030
- 82596DX and MC68020
- 82596SX and MC68000

1.2 Fundamental Assumptions

Each design is based on several fundamental assumptions about the memory subsystem. The circuits required to support these features are implemented in a few programmable components. If the 82596 is added to existing designs in which the required circuits are already implemented, these circuits do not have to be duplicated.

The following assumptions are made about the designs.

- The memory subsystem uses DRAM.
- Refresh request signals are asynchronous to the system clock.
- Interface logic is implemented in PLDs where possible.
- 82596 family type signals will be converted to M68000 family type signals.

2.0 GENERIC IMPLEMENTATION ISSUES

2.1 Block Definitions

Each design is broken into functional blocks. The generic block diagram is shown in Figure 1. The M68000 family and 82596 family are fixed from a functional standpoint. The designer has almost no flexibility in their connection or timing. For the purpose of these designs, the memory control block is also assumed to be fixed. It is set up only for M68000-type signals and timings. The blocks for which there is some design flexibility are grouped under the name control logic. These blocks include the following.

- **Clocking.** Provides the proper clock phases to the 82596 and M68000. It also provides the clock for the other blocks.
- **Reset Retiming.** Takes the active low $\overline{\text{RESET}}$ signal that goes to the M68000 and adjusts its timing and level to be compatible with the active high RESET for the 82596.
- **CA and PORT Generation.** Decodes the address lines and generates the Channel Attention (CA) and CPU Port ($\overline{\text{PORT}}$) signals to the 82596. The system designer selects the memory addresses to be decoded to activate these signals. The amount of decode logic will vary greatly depending on the system memory map.
- **Arbitration.** Determines which of the three master devices has control of the local bus: the M68000, the 82596, or the refresh controller. The refresh controller has the highest priority, followed by the 82596 and then the M68000. Additional master devices are supported through simple changes to the PLD equations in this block.
- **Memory Signal Conversion.** Takes the 82596 control signals, such as $\overline{\text{ADS}}$ and W/$\overline{\text{R}}$, and converts them to M68000-type control signals, such as $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$.
- **Wait State and Burst Generation.** Generates the $\overline{\text{RDY}}$ signal to the 82596 (and $\overline{\text{BRDY}}$ for the 82596CA). It also asserts the burst request ($\overline{\text{CBREQ}}$) to the memory controller.

Table 1. 82596-Family Parallel Bus Comparison

82596 Version	Address Bits	Data Bits	Parallel Clocking	Burst Access	Parity Pins	Maximum Frequency (MHz)
82596CA	32	32	x1	Yes	Yes	33
82596DX	32	32	x2	No	No	33
82596SX	24	16	x2	No	No	20

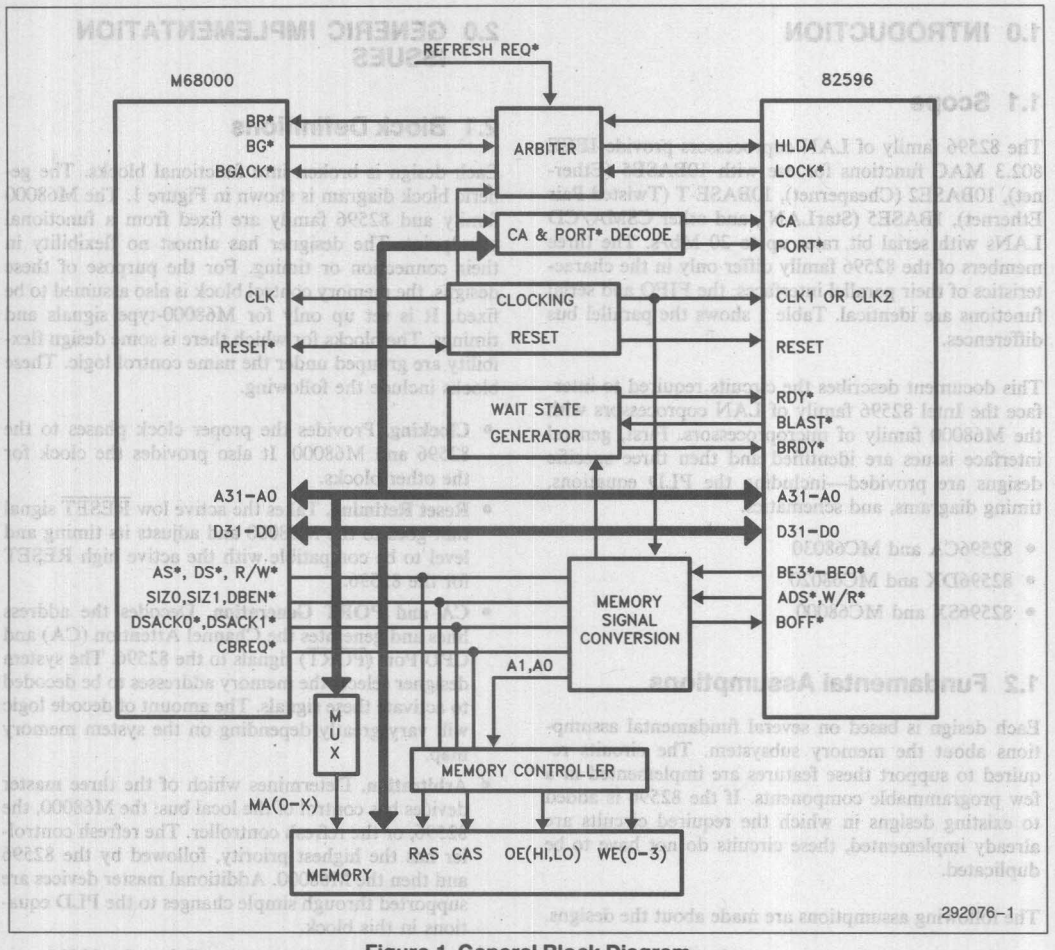


Figure 1. General Block Diagram

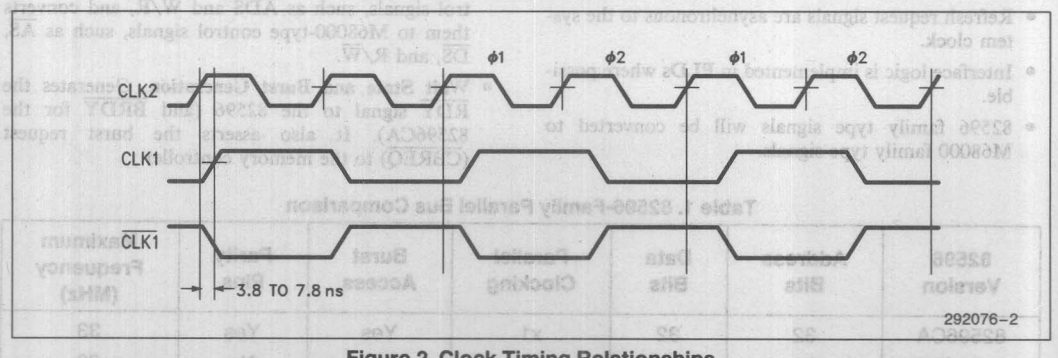


Figure 2. Clock Timing Relationships

2.2 Clocking

The 82596 family uses different types of clocking. The 82596DX and 82596SX use CLK2, which is twice as fast as the internal operating frequency. The two different phases of CLK2 for every CLK1 are defined as $\phi 1$ and $\phi 2$. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$. The 82596CA uses CLK1, which is identical to the internal operating frequency. The 82596 clock timing relationships are shown in Figure 2.

In many cases the control logic is simplified by clocking it with CLK2, even if the CPU and 82596CA are clocked by CLK1 . In some other cases it is advantageous to invert the clocking signal to the 82596, which introduces a phase shift between the devices. The clocking specifications of the 82596 and M68000 are shown in Table 2.

All the designs use 74F74 flip-flops because of their high operating frequency, low propagation delay, and wide availability. If another type of flip-flop is used the timing analysis must be modified to reflect the different specifications.

2.3 Reset Retiming

Because the 82596DX and 82596SX use CLK2, the set-up and hold time specifications for the reset signal are very important. The deactivation of RESET is the only means by which the 82596DX and 82596SX determine which phases of CLK2 correspond to $\phi 1$ and $\phi 2$. Since many of the control signals are only valid at the beginning of certain phases, it is crucial that the arbitration, signal conversion, and wait state generation logic know the current phase of the clock. Failure to meet these specifications can cause improper memory accesses by the 82596. Figure 3 is the reset retiming block schematic.

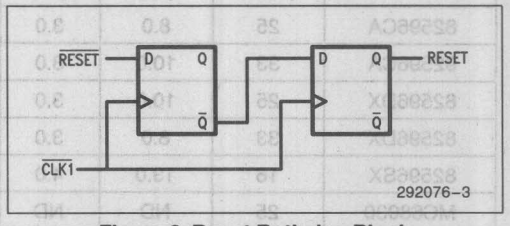


Figure 3. Reset Retiming Block

Table 2. Clocking Specifications

Component	Freq (MHz)	Clock	Max Rise	Max Fall	Min High	Min Low
			(nanoseconds)			
82596CA	25	CLK	3.0	3.0	14.0	14.0
82596CA	33	CLK	3.0	3.0	11.0	11.0
82596DX	25	CLK2	7.0	7.0	4.0	5.0
82596DX	33	CLK2	4.0	3.0	4.5	4.5
82596SX	16	CLK2	8.0	8.0	5.0	7.0
MC68030	25	CLK	4.0	4.0	19.0	19.0
MC68030	33	CLK	3.0	3.0	14.0	14.0
MC68020	25	CLK	4.0	4.0	19.0	19.0
MC68020	33	CLK	3.0	3.0	14.0	14.0
MC68000	16	CLK	5.0	5.0	27.0	27.0

The timings for the M68000 active low signal $\overline{\text{RESET}}$ are not compatible with the 82596 active high signal RESET. To prevent possible metastable conditions, the M68000 $\overline{\text{RESET}}$ passes through a two-stage synchronizer before going to the 82596. This will usually require two 74F74 flip-flops. Using two stages, rather than one, greatly reduces the probability of metastable conditions in the 82596. One of the stages is also used to invert the signal. Table 3 lists the relevant specifications for RESET and $\overline{\text{RESET}}$.

Table 3. Reset Specifications

Component	Freq (MHz)	Setup (ns)	Hold (ns)
82596CA	25	8.0	3.0
92596CA	33	10.0	3.0
82596DX	25	10.0	3.0
82596DX	33	8.0	3.0
82596SX	16	13.0	4.0
MC68030	25	ND	ND
MC68030	33	ND	ND
MC68020	25	ND	ND
MC68020	33	ND	ND
MC68000	16	ND	ND

ND = Not Defined by Motorola

2.4 CA and PORT Generation

The 82596 has two inputs that do not correspond to any signals generated by the CPU: Channel Attention (CA) and CPU Port ($\overline{\text{PORT}}$). Channel Attention is always monitored by the 82596, and the falling edge is internally latched. The 82596 responds to Channel Attention by reading the system control block command word, which is stored in memory. Several fields in this command word tell the 82596 what to do; e.g., acknowledge interrupts, change the state of the command unit, change the state of the receive unit, or load the bus throttle timers.

When the 82596 does not have the bus, it examines $\overline{\text{PORT}}$. If it is active the value on the data bus is stored in the 82596 in a special register. The value on the 4 least significant bits (D3–D0) indicates one of sixteen functions. Only four functions are defined (functions 4 through 15 are reserved and should not be used).

0 Do an internal reset (pins D31–D4 are ignored).

1 Do a self test (the results are placed at the location specified by pins D31–D4).

2 Execute a Dump command (the results are placed at the location specified by pins D31–D4).

3 Move the system configuration pointer to the location specified on pins D31–D4.

$\overline{\text{PORT}}$ has more stringent requirements for setup and hold times than CA does. $\overline{\text{PORT}}$ must meet specific setup and hold times with respect to the clock and must also be active for at least two consecutive clocks. CA is only required to be active for two clocks without meeting specific setup or hold times; however, CA only has to be active for one clock if the setup and hold times are met.

Because the M68000 family does not support a separate I/O address space, all I/O functions must be memory-mapped . The addresses for CA and $\overline{\text{PORT}}$ can be selected by the designer. In the design examples, Section 3 through 5, part of one PLD is used to generate both signals. The number of input pins on the PLD will determine the address range limitations. If the PLD has fewer input pins than the number of address lines to be decoded, one of the PLD inputs should be connected to the output of a secondary decoder. This secondary decoder must meet a worst-case propagation delay, which is listed in the comment fields of the PLD equations for each design.

Implementing CA and $\overline{\text{PORT}}$ will usually require four macro-cells, which is about one-half of a standard PLD. Two are used for the actual output signals and two are used as a state machine to control the timing of the output signals. Figure 4 is the CA and PORT generator block diagram and Figure 5 is the state transition diagram.

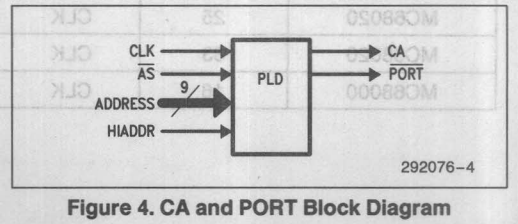


Figure 4. CA and PORT Block Diagram

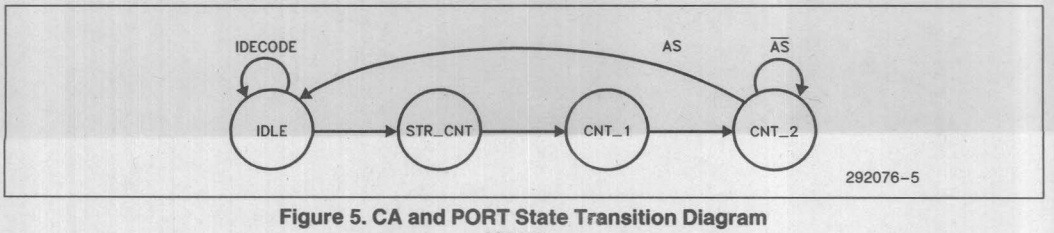


Figure 5. CA and PORT State Transition Diagram

2.5 Arbitration

All the design examples assume that the memory will use DRAMs. This means there will be at least three master devices attempting to gain access to memory: the M68000 CPU, the 82596 LAN coprocessor, and the refresh controller. The requests from the refresh controller must be given highest priority to avoid corrupting data in the DRAMs. The 82596 is given the second highest priority, so it is not forced to wait and eventually overrun or underrun. The M68000 has the lowest priority because of its internal Bus Request/Bus Grant mechanism. Because some of the M68000 family CPUs have an internal cache or instruction pipeline, they can fetch code or data internally while the 82596 or the refresh controller are using the local bus.

The M68000 family uses a three-signal arbitration scheme. A master device makes a bus request by asserting $\overline{BR}$ and waiting for the M68000 to assert $\overline{BG}$. The master device then drives $\overline{BGACK}$ while it is using the bus. When the master device no longer needs the bus it brings $\overline{BR}$ inactive, then the M68000 drives $\overline{BG}$ inactive. Finally the master device drives $\overline{BGACK}$ inactive.

2.5.1 REFRESH REQUESTS

Refresh requests are assumed to be asynchronous with the arbitration clock; therefore, the refresh signal must be synchronized—typically with a 74F74 flip-flop. At the completion of the refresh cycle the local bus will be released to the requestor having highest priority. The flip-flop is not needed if the refresh request timing is synchronous with the arbitration clock and meets the setup and hold times of the PLD.

If a refresh request arrives while the M68000 is the active bus master, the Bus Request signal ($\overline{BR}$) to the M68000 will be asserted. When the M68000 forces $\overline{BG}$ active the arbitration logic brings $\overline{BGACK}$ active and the refresh cycle begins. When the refresh has been completed $\overline{BR}$ goes inactive. If the 82596 is the active bus master when the refresh request arrives, the refresh cycle will not start until the 82596 has completed its transfers. $\overline{BR}$ to the M68000 will remain active until the refresh cycle has completed; $\overline{BR}$ will not deassert when the 82596 completes its transfers. If another 82596 request arrives during the refresh cycle, $\overline{BR}$ will remain active until both the refresh controller and the 82596 complete their transfers.

The designer is responsible for ensuring that enough refresh requests are made to avoid corrupting data in the DRAM. These designs assume that a refresh cycle signal goes into the memory controller and indicates that a refresh cycle is in progress. If a transparent technique is used for refreshing the DRAM, or if SRAM is used, then the arbiter can be greatly simplified.

There are several transparent DRAM refresh techniques. The most common method hides the refresh cycle as extra wait states in the normal CPU or 82596 accesses. This technique eliminates the arbitration overhead of the $\overline{BR}/\overline{BG}$ (HOLD/HLDA) protocol and simplifies the arbiter logic. The main disadvantage is that the wait state generator becomes more complex.

2.5.2 82596 REQUESTS

The 82596 acquires and holds the system bus via the HOLD/HLDA handshake. It requests the bus by activating HOLD. When the arbiter gives the local bus to the 82596 it asserts the HLDA signal, which is the inverted $\overline{LANCYC}$ signal from the arbiter. Overrun conditions can occur in some external devices if the 82596 holds the bus too long. The 82596's bus throttle timers can be used to regulate bus use; the timer can be activated two ways.

- **Externally.** A high state on the BREQ pin starts the timer.
- **Internally.** A high state on the HLDA pin starts the timer.

Instead of using bus arbitration schemes, the 82596CA can be forced off the bus by activating the backoff pin ($\overline{BOFF}$). This provides higher performance and faster refresh cycles. (The 82596DX and 82596SX do not have this backoff feature.)

Because the 82596 HOLD and HLDA signals are active high and the M68000 $\overline{BR}$ and $\overline{BG}$ are active low, the arbiter must invert the logic. In addition, the timings are not compatible. The arbitration signal timings are shown in Tables 4 and 5.

Table 4. Arbitration Signal Input Timings

Component	Frequency (MHz)	Signal	Output Valid Delay (ns)
82596CA	25	HOLD	3 to 22
82596CA	33	HOLD	3 to 19
82596DX	25	HOLD	4 to 22
82596DX	33	HOLD	3 to 19
82596SX	16	HOLD	4 to 32
MC68030	25	$\overline{BG}$	0 to 20
MC68030	33	$\overline{BG}$	0 to 20
MC68020	25	$\overline{BG}$	0 to 20
MC68020	33	$\overline{BG}$	0 to 20
MC68000	16	$\overline{BG}$	0 to 40

2.5.3 ARBITER IMPLEMENTATION

Local bus arbitration is mostly implemented in a synchronous PLD that uses the inverted CPU clock ($\overline{CLK1}$) as the arbitration clock. The arbiter has fixed priorities and responds to bus requests from the 82596 and the refresh controller by requesting the local bus from the M68000. The arbiter asserts the Bus Request ($\overline{BR}$) and Bus Grant Acknowledge ($\overline{BGACK}$) signals to the M68000, and enforces the bus arbitration protocol.

The arbiter does not immediately give the bus to the requestor. The arbiter is usually required to provide an adequate DRAM precharge time and will not release the bus until the precharge time has expired. The arbiter can be greatly simplified if other logic is used to control the precharge time. Figure 6 is the arbiter state transition diagram and the signal timings are shown in Figure 7.

Table 5. Arbitration Signal Output Timings

Component	Frequency (MHz)	Signal	Minimum Setup (ns)	Minimum Hold (ns)
82596CA	25	HLDA	10	3
82596CA	33	HLDA	8	3
82596DX	25	HLDA	10	3
82596DX	33	HLDA	8	3
82596SX	16	HLDA	11	8
MC68030	25	$\overline{BR}$	NA	NA
MC68030	25	$\overline{BGACK}$	NA	NA
MC68030	33	$\overline{BR}$	NA	NA
MC68030	33	$\overline{BGACK}$	NA	NA
MC68020	25	$\overline{BR}$	NA	NA
MC68020	25	$\overline{BGACK}$	NA	NA
MC68020	33	$\overline{BR}$	NA	NA
MC68020	33	$\overline{BGACK}$	NA	NA
MC68000	16	$\overline{BR}$	NA	NA
MC68000	16	$\overline{BGACK}$	NA	NA

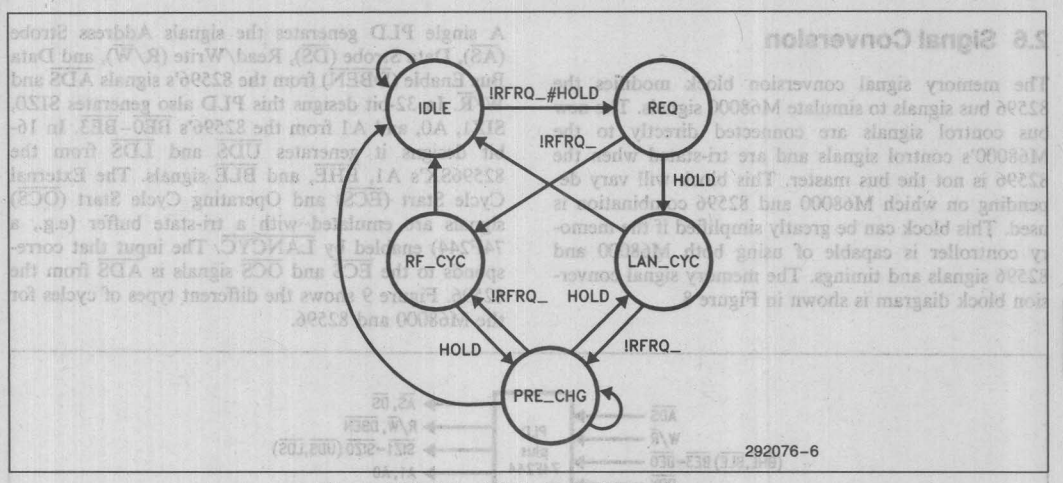


Figure 6. Arbiter State Transition Diagram

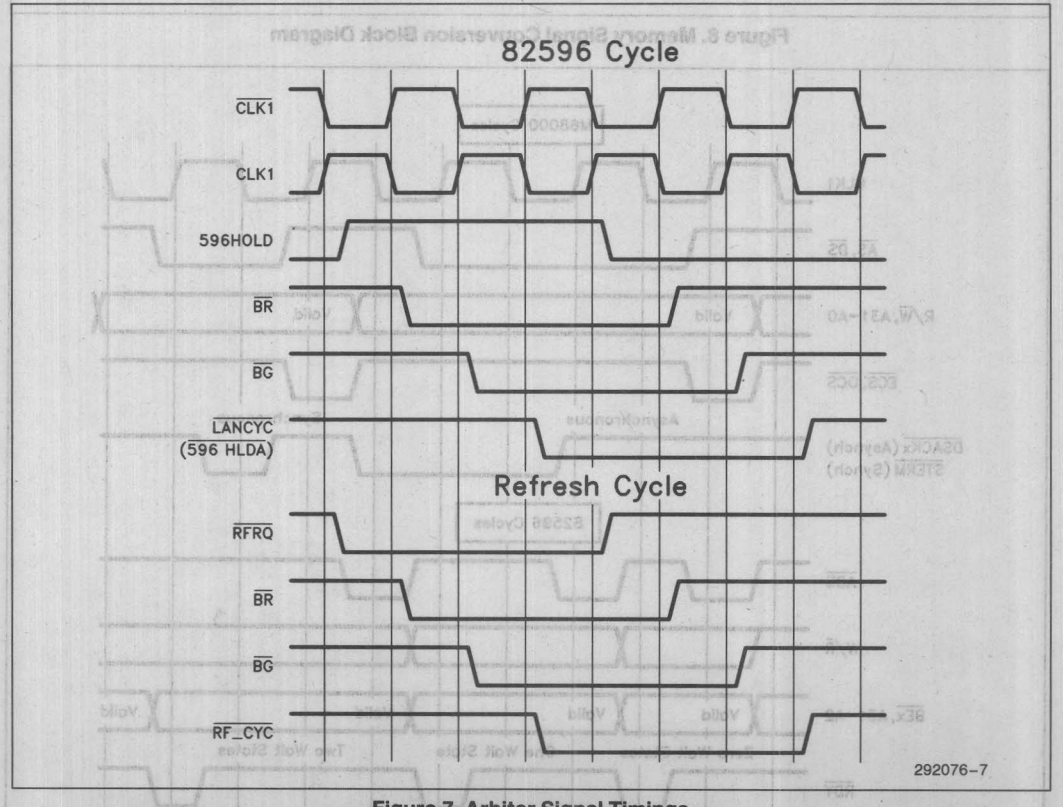


Figure 7. Arbiter Signal Timings

2.6 Signal Conversion

The memory signal conversion block modifies the 82596 bus signals to simulate M68000 signals. The new bus control signals are connected directly to the M68000's control signals and are tri-stated when the 82596 is not the bus master. This block will vary depending on which M68000 and 82596 combination is used. This block can be greatly simplified if the memory controller is capable of using both M68000 and 82596 signals and timings. The memory signal conversion block diagram is shown in Figure 8.

A single PLD generates the signals Address Strobe ($\overline{AS}$), Data Strobe ($\overline{DS}$), Read/Write (R/$\overline{W}$), and Data Bus Enable ($\overline{DBEN}$) from the 82596's signals $\overline{ADS}$ and W/$\overline{R}$. In 32-bit designs this PLD also generates SIZ0, SIZ1, A0, and A1 from the 82596's $\overline{BE0}$–$\overline{BE3}$. In 16-bit designs it generates $\overline{UDS}$ and $\overline{LDS}$ from the 82596SX's A1, $\overline{BHE}$, and $\overline{BLE}$ signals. The External Cycle Start ($\overline{ECS}$) and Operating Cycle Start ($\overline{OCS}$) signals are emulated with a tri-state buffer (e.g., a 74F244) enabled by $\overline{LANCYC}$. The input that corresponds to the $\overline{ECS}$ and $\overline{OCS}$ signals is $\overline{ADS}$ from the 82596. Figure 9 shows the different types of cycles for the M68000 and 82596.

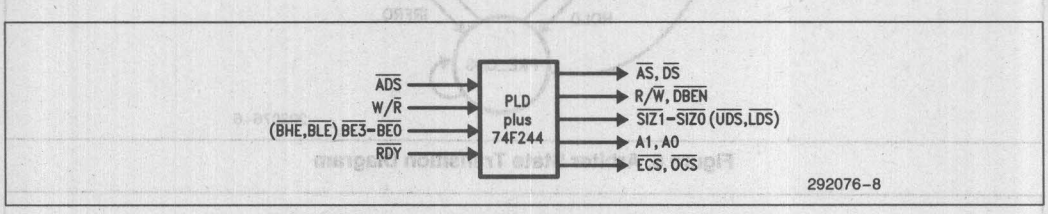


Figure 8. Memory Signal Conversion Block Diagram

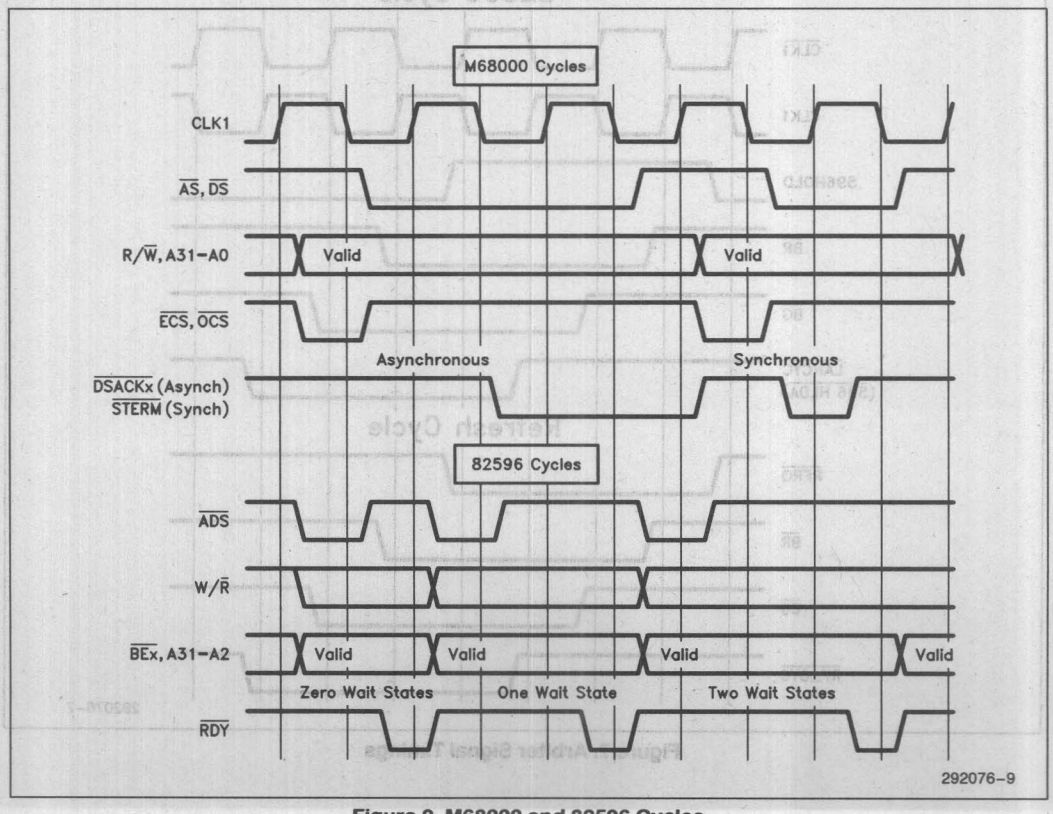


Figure 9. M68000 and 82596 Cycles

2.7 Wait State and Burst Generator

2.7.1 GENERAL INFORMATION

The 82596 and M68000 combinations can use three types of bus transfers (the 82596DX and 82596SX support only a basic single-cycle transfer). More than one single transfer can occur without interruption. Each transfer requires at least two clocks and begins with $\overline{ADS}$ going active during the first clock cycle and then $\overline{RDY}$ goes active in the last clock. Wait states are inserted by keeping $\overline{RDY}$ inactive. Figure 10 shows the wait state and burst generator block diagram.

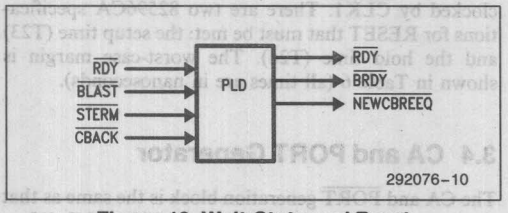


Figure 10. Wait-State and Burst Generator Block Diagram

The 82596CA supports all three types of transfers: single cycle, multiple cycle, and burst. The multiple cycle is simply several uninterrupted single-cycle transfers, with each bus cycle beginning with $\overline{ADS}$ going active during the first clock and then $\overline{RDY}$ going active in the last clock. Burst cycles can contain as many as four consecutive data transfers to four consecutive locations; however, $\overline{ADS}$ is only generated before the first data transfer. The maximum amount of data moved during a burst is 16 bytes (four 4-byte transfers). The wait state and burst generation block inserts the appropriate number of wait states during the bus cycle by driving $\overline{RDY}$ and $\overline{BRDY}$ active at the appropriate time.

The M68000 family supports a 3-clock asynchronous cycle. The MC68030 also supports a 2-clock synchronous cycle that is similar to the 82596CA burst access.

2.7.2 SINGLE-CYCLE BUS TRANSFERS

The 82596 begins a cycle after the rising edge of CLK2 (CLK for the 82596CA) by asserting $\overline{ADS}$ and driving W/$\overline{R}$ and the address lines (A31–A2) valid. The conversion PLD synchronizes $\overline{ADS}$ to the clock and generates an address strobe ($\overline{NEWAS}$). $\overline{NEWAS}$ is asserted during the same phase of the clock that a M68000 would assert $\overline{AS}$. $\overline{NEWAS}$ is also asserted during the second clock of the 82596 transfer.

The wait state generator delays the $\overline{RDY}$ signal to the 82596 from going active. This provides time to meet the data setup and hold specifications. The 82596 requires that data be set up a few nanoseconds before the rising edge of its clock. The M68000 also requires a setup time; however, it is usually only one nanosecond. The system designer will need to make provision for, at the least, the 82596 data setup time plus an additional 2 ns for clock skew. If this cannot be met another wait state will be needed for all 82596 memory read cycles. This can be provided by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks at the end of a read cycle. The 82596CA asserts Burst Last ($\overline{BLAST}$) during the second clock of the first cycle, which indicates that the transfer is complete after a single cycle (the 82596DX and 82596SX do not use $\overline{BLAST}$).

2.7.3 BURST-CYCLE AND MULTIPLE-CYCLE BUS TRANSFERS

The 82596CA tries to burst cycles for any bus request that requires more than a single data cycle to consecutive addresses. The starting address must begin on an 8-byte boundary (xxxxxxx0h or xxxxxxx8h). The fastest burst cycle for this design assumes that 80 ns interleaved DRAMs are being used, which are fast enough to allow new data to be strobed into the 82596CA on each clock. The burst cycle requires 4 clocks for the first data strobe; however, subsequent data strobes are returned with each clock.

Burst cycles begin with the 82596CA driving a valid address and asserting $\overline{ADS}$ in the same manner as non-burst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding $\overline{BLAST}$ inactive during the second clock of the cycle. The ready logic then generates a Cache Burst Request ($\overline{NEWCBREQ}$) signal to the memory controller. If Cache Burst Acknowledge ($\overline{CBACK}$) is returned active it indicates that the memory can operate in burst mode. Then the ready logic waits for the Synchronous Termination ($\overline{STERM}$) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. When $\overline{STERM}$ is received the ready logic activates $\overline{BRDY}$ to the 82596CA, which indicates its willingness to allow a burst cycle. The 82596CA drives $\overline{BLAST}$ inactive for all but the last cycle in a burst. $\overline{BLAST}$ is driven active in the last cycle of the transfer to indicate that when either $\overline{BRDY}$ or $\overline{RDY}$ is next returned the transfer is complete. $\overline{RDY}$ is always returned in response to $\overline{BLAST}$ going active.

If the memory controller cannot perform a burst cycle $\overline{CBACK}$ will not go active and the ready logic will return $\overline{RDY}$ to the 82596CA, which indicates a non-burst multiple-cycle transfer will take place. Unlike the burst cycle, $\overline{ADS}$ will go active at the beginning of the second and all subsequent transfers in the multicycle transfer and $\overline{RDY}$ is used to end the cycle rather than $\overline{BRDY}$.

The two data acknowledge signals for the MC68030 ($\overline{DSACK0}$ and $\overline{DSACK1}$) can be combined because the 82596CA only needs one $\overline{RDY}$ signal. Both $\overline{DSACK}$ signals connect to the inputs of an 74F08 AND gate.

3.0 MC68030/82596CA INTERFACE

3.1 Design Specifications

This interface example is based on the following assumptions.

- MC68030 CPU.
- 82596CA LAN coprocessor.
- 32-bit DRAM memory with burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 and 25 MHz operating frequencies.
- Interface logic implemented in PLDs where possible.
- 82596CA signals converted to MC68030 signal types.
- Refresh request signal asynchronous to 33 MHz clock.
- Burst accesses attempted whenever possible.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68030. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

3.2 Clocking

This design uses a clock operating at 66 MHz. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68030 uses CLK1, but the 82596CA and arbitration logic use $\overline{CLK1}$. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns.

3.3 Reset Retiming

The 82596CA reset retiming block is the same as that shown in Figure 2. The synchronizing flip-flops are clocked by $\overline{CLK1}$. There are two 82596CA specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 6 (all times are in nanoseconds).

3.4 CA and PORT Generator

The CA and $\overline{PORT}$ generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed it is clocked by $\overline{CLK1}$. $\overline{AS}$, the address lines, and HIADDR are examined at the rising edge of $\overline{CLK1}$. The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because $\overline{AS}$ is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

$$\overline{CLK1}\text{ cycle} - \text{max MC68030 }\overline{AS}\text{ valid delay} - \text{max }\overline{CLK1}\text{ to CLK1 skew} - \text{min PLD setup} \quad (3.1)$$

At 33 MHz = 30 − 15 − 2 − 10 = 3 ns (with 10 ns PLD)

At 25 MHz = 40 − 18 − 2 − 15 = 5 ns (with 15 ns PLD)

Table 6. 82596CA Worst-Case Reset Timing Margin

82596CA Frequency (MHz)	Clock-to-Output Delay		Minimum Setup	Minimum Hold	Margin	
	Minimum	Minimum			Setup	Hold
	(nanoseconds)					
33	3.8	7.8	8.0	3.0	14.2	0.8
25	3.8	7.8	10.0	3.0	22.2	0.8

The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

$$\overline{CLK1}\text{ cycle} + \overline{CLK1}\text{ high time} - \text{max MC68030 address valid delay} - \text{max } \overline{CLK1}\text{ to CLK1 skew} - \text{min PLD setup} \quad (3.2)$$

At 33 MHz = 30 + 15 − 21 − 2 − 10 = 12 ns (with 10 ns PLD)

At 25 MHz = 40 + 20 − 25 − 2 − 15 = 18 ns (with 15 ns PLD)

Next, the worst-case setup and hold times to the 82596CA are calculated for CA and $\overline{PORT}$, which have identical timings. They go active based on the rising edge of $\overline{CLK1}$. The setup margins are calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max PLD output valid delay} - \text{min 82596CA input setup} \quad (3.3)$$

At 33 MHz = 30 − 7 − 7 = 16 ns (with 10 ns PLD)

At 25 MHz = 40 − 12 − 7 = 21 ns (with 15 ns PLD)

The hold margins are calculated as follows.

$$\text{min PLD output valid delay} - \text{min 82596CA input hold} = 4 - 3 = 1\text{ ns (at 33 and 25 MHz)} \quad (3.4)$$

3.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz). At either speed, it is clocked by $\overline{CLK1}$. The worst-case flip-flop setup margins to this rising edge is calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max 82596CA HOLD output valid} - \text{min PLD input setup} \quad (3.5)$$

At 33 MHz = 30 − 19 − 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 − 22 − 15 = 3 ns (with 15 ns PLD)

The signal $\overline{BR}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68030. The PLD flip-flop setup time for $\overline{BG}$ is checked next. Because $\overline{BG}$ can go active 0 ns after the falling edge of CLK1, and because there can be up to 2 ns of skew between CLK1 and $\overline{CLK1}$, it is not completely safe to directly use $\overline{BG}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{BG}$ can go active about the same time as $\overline{CLK1}$ goes high. Because the PLD will not be clocked until the next rising edge of $\overline{CLK1}$, there will be at least one full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The unused macro-cell in the 20R8 can be used to invert $\overline{LANCYC}$ to create HLDA to the 82596CA. The outputs of the arbiter, HLDA and $\overline{REFCYC}$, are internally synchronized at their destination, so no output timing analysis is required.

3.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596CA-type control signals to MC68030-type control signals. When the 82596CA does not have the bus all the outputs go to a high-impedance state.

The signals SIZ1, SIZ0, A1, and A0 are generated by using simple combinatorial decodes of $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$, and $\overline{BE0}$. The delay will be identical to the PLD propagation delay. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by $\overline{CLK1}$. Their states are determined by the state of the 82596CA's signals $\overline{ADS}$, W/$\overline{R}$, and $\overline{RDY}$.

An 82596CA read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of $\overline{CLK1}$. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of $\overline{CLK1}$. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596CA control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\overline{CLK1}\text{ cycle} - \text{max 82596CA output delay} - \text{PLD input setup} \quad (3.6)$$

At 33 MHz = 30 − 19 − 10 = 1 ns (with 10 ns PLD)

At 25 MHz = 40 − 22 − 15 = 3 ns (with 15 ns PLD)

NEWAS goes active based on the rising edge of CLK1, which is the same as the falling edge of CLK1. The PLD clock to output valid delay is 3 to 7 ns maximum at 33 MHz and 4 to 12 ns maximum at 25 MHz. The skew between the clocks will be −2 to +2 ns. This translates to a 1 to 5 ns delay at 33 MHz and 2 to 10 ns delay at 25 MHz, which is within the MC68030 specifications of 2 to 10 ns.

ECS and OCS are generated by taking ADS and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596CA has the bus ECS and OCS will go active about 4 to 8 ns after ADS goes active.

3.7 Wait State and Burst Generator

3.7.1 GENERAL INFORMATION

The 82596CA supports three types of bus transfers: single cycle, multiple cycle, and burst. Each bus cycle is at least two clocks long and begins with ADS going active during the first clock and RDY active in the last clock. A bus cycle contains one or more data transfers, each of which can be up to 32 bits. Burst cycles can contain as many as four data transfers, thus, the maximum amount of data moved during a burst is 16 bytes (4 transfers of 4 bytes each). The wait state and burst generation block inserts the proper number of wait states during the bus cycle. For this design it was assumed that the DRAM would allow for zero wait state accesses for the second through fourth data transfers during a burst cycle. If slower DRAMs are used, wait states will need to be inserted in the DSACK and RDY generation circuits.

3.7.2 SINGLE CYCLE TRANSFERS

The fastest single cycle transfer in this design requires three clocks for the 82596CA. The 82596CA initiates a cycle after the rising edge of CLK1 by asserting ADS and driving W/R and the address lines (A31–A2) valid. The conversion PLD synchronizes ADS and generates an address strobe (NEWAS). NEWAS is asserted during the same phase of the clock that a MC68030 would assert AS. NEWAS is also asserted during the second clock of the 82596CA transfer.

The wait state generator delays the RDY signal to the 82596CA. This provides enough time to meet the data setup and hold specifications. The 82596CA requires that data be valid at least 5 ns before the rising edge of its clock. The MC68030 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596CA has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596 memory read cycles. This can be done by modifying the PLD equations to delay the assertion of RDY by one or more clocks. The 82596CA asserts Burst Last (BLAST) during the second clock of the first cycle, which indicates that the transfer is complete after a single cycle.

3.7.3 BURST CYCLE BUS TRANSFERS

The 82596CA attempts burst cycles for any bus request that requires more than a single data cycle to consecutive addresses. The starting address must begin on an 8-byte boundary (xxxxxxx0h or xxxxxxx8h). The fastest burst cycle for this design assumes 80 ns interleaved DRAMs, which allow new data to be strobed into the 82596CA on each clock. The burst cycle requires four clocks for the first data strobe, but subsequent data strobes are returned with each clock.

Burst cycles begin with the 82596CA driving a valid address and asserting ADS in the same manner as nonburst cycles. The 82596CA indicates that it is willing to enter a burst cycle by holding BLAST inactive in the second clock of the cycle. The ready logic then generates a cache burst request (NEWCBREQ) signal to the memory controller. If the cache burst acknowledge signal (CBACK) is returned active it indicates that the memory can operate in burst mode. The ready logic then waits for the synchronous termination (STERM) bus handshake signal, which indicates that the correct number of wait states has occurred and data is valid. The ready logic then activates BRDY to the 82596CA to indicate its willingness to permit a burst cycle. The 82596CA drives BLAST inactive for all but the last cycle in a burst. BLAST is driven active in the last cycle of the transfer to indicate that when RDY or BRDY is next returned the transfer is complete. RDY is always returned in response to BLAST going active.

If the memory controller cannot perform a burst cycle CBACK will not go active and the ready logic will return RDY to the 82596CA to indicate a nonburst multiple-cycle transfer will take place. This bus transfer is simply a sequence of two or more single cycle transfers. Unlike the burst cycles, ADS goes active during the first clock of the second through fourth data transfers. The timing margins for these cycles are identical to those for nonburst single cycle transfers.

Because the 82596CA requires only one RDY signal, the two data acknowledge signals for the MC68030 (DSACK0 and DSACK1) can be combined. Both DSACK signals connect to the inputs of an 74F08 AND gate.

4.0 MC68020/82596DX INTERFACE

4.1 Design Specifications

This interface example is based on the following assumptions.

- MC68020 CPU.
- 82596DX LAN coprocessor.
- 32-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 33 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596DX signals converted to MC68020 signal types.
- Refresh request signal asynchronous to 33 MHz clock.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68020. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

4.2 Clocking

This design uses a clock operating at 66 MHz. The 66 MHz clock, CLK2, is directly by the 82596DX. It is divided by a 74F74 flip-flop to generate two 33 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68020 uses $\overline{CLK1}$, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$.

4.3 Reset Retiming

The 82596DX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596DX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7.

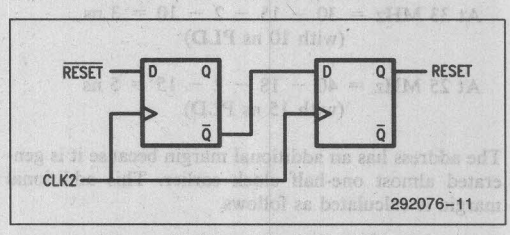


Figure 11. Reset Retiming Block

Table 7. 82596DX/SX Worst-Case Reset Timing Margin

82596 Frequency (MHz)	Clock-to-Output Delay		Minimum Setup	Minimum Hold	Margin	
	Minimum	Maximum			Setup	Hold
	(nanoseconds)					
33	3.8	7.8	8.0	3.0	14.2	0.8
25	3.8	7.8	10.0	3.0	22.2	0.8
16	3.8	7.8	13.0	4.0	45.2	−0.2

4.4 CA and PORT Generator

The CA and $\overline{\text{PORT}}$ generation block is the same as that shown in Figure 4 and is based on a 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. $\overline{\text{AS}}$, the address lines, and HIADDR are examined at the rising edge of CLK1. The worst-case margin to this rising edge limits the maximum propagation delay of the secondary decoder. Each margin is calculated separately.

Because $\overline{\text{AS}}$ is generated later than the address, it is checked first. The setup time margin to the PLD's flip-flop is calculated as follows (all times are in nanoseconds).

$$\begin{aligned}\text{CLK1 cycle} &- \text{max MC68020 } \overline{\text{AS}} \text{ valid delay} \\ &- \text{max CLK1 to } \overline{\text{CLK1}} \text{ skew} \\ &- \text{min PLD setup}\end{aligned} \quad (4.1)$$

$$\text{At 33 MHz} = 30 - 15 - 2 - 10 = 3 \text{ ns} \quad \text{(with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 - 18 - 2 - 15 = 5 \text{ ns} \quad \text{(with 15 ns PLD)}$$

The address has an additional margin because it is generated almost one-half clock earlier. This additional margin is calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &+ \text{CLK1 low time} \\ &- \text{max MC68020 address valid delay} \\ &- \text{max } \overline{\text{CLK1}} \text{ to CLK1 skew} \\ &- \text{min PLD setup}\end{aligned} \quad (4.2)$$

$$\text{At 33 MHz} = 30 + 15 - 21 - 2 - 10 = 12 \text{ ns} \quad \text{(with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 + 20 - 25 - 2 - 15 = 18 \text{ ns} \quad \text{(with 15 ns PLD)}$$

Next, the worst-case setup and hold times to the 82596DX are calculated for CA and $\overline{\text{PORT}}$, which have identical timings. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &- \text{max PLD output valid delay} \\ &- \text{min 82596DX input setup} \\ &- \text{max CLK2 to CLK1 clock skew}\end{aligned} \quad (4.3)$$

$$\text{At 33 MHz} = 30 - 7 - 7 - 7.8 = 8.2 \text{ ns} \quad \text{(with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 - 12 - 7 - 7.8 = 13.2 \text{ ns} \quad \text{(with 15 ns PLD)}$$

The hold margins are calculated as follows.

$$\begin{aligned}&\text{PLD output valid delay - min 82596DX input hold} \\ &+ \text{min CLK2 to CLK1 skew} \\ &= 4 - 3 + 3.8 = 4.8 \text{ ns (at 33 and 25 MHz)}\end{aligned} \quad (4.4)$$

4.5 Bus Arbiter

The bus arbiter is implemented with a 20R8 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

$$\begin{aligned}\text{CLK1 cycle} &+ \text{min CLK2 to CLK1 skew} \\ &- \text{max 82596DX HOLD output valid} \\ &- \text{min PLD input setup}\end{aligned} \quad (4.5)$$

$$\text{At 33 MHz} = 30 + 3.8 - 19 - 10 = 4.8 \text{ ns} \quad \text{(with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 + 3.8 - 22 - 15 = 7.8 \text{ ns} \quad \text{(with 15 ns PLD)}$$

The signal $\overline{\text{BR}}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68020. The PLD flip-flop setup time for $\overline{\text{BG}}$ is checked next. Because $\overline{\text{BG}}$ can go active 0 ns after the falling edge of $\overline{\text{CLK1}}$, and because there can be up to 2 ns of skew between CLK1 and $\overline{\text{CLK1}}$, it is not completely safe to directly use $\overline{\text{BG}}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{\text{BG}}$ can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be a full clock cycle minus the PLD feedback delay for the output to reach a valid state.

The outputs of the arbiter, $\overline{\text{LANCYC}}$ and $\overline{\text{REFCYC}}$, are internally synchronized at their destination, so no output timing analysis is required. An external inverter is required for $\overline{\text{LANCYC}}$ to create HLDA to the 82596DX. If the PLD has an internal inverter then this will not be required.

4.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4 PLD (10 ns delay at 33 MHz, 15 ns delay at 25 MHz) is used to convert the 82596DX-type control signals to MC68020-type control signals. When the 82596DX does not have the bus all the outputs go to a high-impedance state.

The signals SIZ1, SIZ0, A1, and A0 are generated by using simple combinatorial decodes of $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$, and $\overline{BE0}$. The delay will be identical to the PLD propagation delay. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596DX's signals $\overline{ADS}$, W/$\overline{R}$, and $\overline{RDY}$.

An 82596DX read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of CLK1. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of CLK1. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596DX control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\text{CLK1 cycle} + \text{min CLK2 to CLK1 skew} - \text{max 82596DX output delay} - \text{PLD input setup} \quad (4.6)$$

$$\text{At 33 MHz} = 30 + 3.8 - 19 - 10 = 4.8 \text{ ns (with 10 ns PLD)}$$

$$\text{At 25 MHz} = 40 + 3.8 - 22 - 15 = 7.8 \text{ ns (with 15 ns PLD)}$$

$\overline{NEWAS}$ goes active based on the rising edge of CLK1, which is the same as the falling edge of $\overline{CLK1}$. The PLD clock to output valid delay is 2 to 7 ns maximum. The skew between the clocks will be −2 to +2 ns. This translates to a 0- to 5 ns delay, which is within the MC68020 specifications of 2 to 10 ns.

$\overline{ECS}$ and $\overline{OCS}$ are generated by taking $\overline{ADS}$ and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596DX has the bus $\overline{ECS}$ and $\overline{OCS}$ will go active about 8 ns after $\overline{ADS}$.

4.7 Wait State Generator

Each 82596DX bus cycle is at least two clocks long and begins with $\overline{ADS}$ going active during the first clock and $\overline{RDY}$ active in the last clock. The wait state block inserts the proper number of wait states during the bus cycle by delaying the $\overline{RDY}$ signal to the 82596DX. The fastest single transfer in this design requires three clocks for the 82596DX. This provides enough time to meet the data setup and hold specifications. The 82596DX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68020 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596DX has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596DX memory read cycles. This can be done by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks.

Because the 82596DX requires only one $\overline{RDY}$ signal, the two data acknowledge signals for the MC68020 ($\overline{DSACK0}$ and $\overline{DSACK1}$) can be combined. Both $\overline{DSACK}$ signals connect to the inputs of an 74F08 AND gate.

5.0 MC68000/82596SX INTERFACE

5.1 Design Specifications

This interface example is based on the following assumptions.

- MC68000 CPU.
- 82596SX LAN coprocessor.
- 16-bit DRAM memory without burst capability.
- DRAM refresh using CAS-before-RAS technique.
- 16 MHz operating frequency.
- Interface logic implemented in PLDs where possible.
- 82596SX signals converted to MC68000 signal types.
- Refresh request signal asynchronous to 16 MHz clock.

NOTE:
Many of the circuit elements (e.g., PLDs and flip-flops) in this design probably already exist in designs presently using the MC68000. The extra elements are provided only for completeness. The final design will probably require fewer circuit elements.

5.2 Clocking

This design uses a clock operating at 32 MHz. The 32 MHz clock, CLK2, is used directly by the 82596SX. It is divided by a 74F74 flip-flop to generate two 16 MHz clocks from the Q and $\overline{Q}$ outputs: CLK1 and $\overline{CLK1}$. The MC68000 uses $\overline{CLK1}$, but the arbitration logic uses CLK1. The clock-to-output-valid delay of the 74F74 is 3.8 to 7.8 ns. The rising edge of CLK1 corresponds to the rising edge of CLK2 at the beginning of $\phi 1$.

5.3 Reset Retiming

The 82596SX reset retiming block is shown in Figure 11. The synchronizing flip-flops are clocked by CLK2. There are two 82596SX specifications for RESET that must be met: the setup time (T23) and the hold time (T24). The worst-case margin is shown in Table 7 (all times are in nanoseconds).

5.4 CA and PORT Generator

The CA and $\overline{PORT}$ generation block is the same as that shown in Figure 4 and is based on a 20R4-15 PLD clocked by CLK1. $\overline{AS}$, the address lines, HIADDR, $\overline{LDS}$, and CLK1 are decoded in a combinatorial macro-cell of the 20R4. The macro-cell output is sent to the input of one of the registered macro-cells, which is clocked at the rising edge of CLK1. Since propagation delay through the PLD is much less than the CLK1 cycle time, there will be a large margin on the flip-flop setup time.

Next, the worst-case setup and hold times to the 82596SX are calculated. The 82596SX timings are identical for both CA and $\overline{PORT}$. They go active based on the rising edge of CLK1. The setup margins are calculated as follows.

$$\begin{aligned} \text{CLK1 cycle} &- \text{max PLD output} \\ &- \text{min 82596SX input setup} \\ &- \text{max CLK2 to CLK1 skew} \\ &= 66 - 12 - 11 - 7.8 = 35.2 \text{ ns} \end{aligned} \quad (5.1)$$

Margins are calculated as follows.

$$\begin{aligned} &- \text{min PLD output valid delay} \\ &- \text{min 82596SX input hold} \\ &+ \text{max CLK2 to CLK1 skew} \\ &= 4 - 6 + 3.8 = 1.8 \text{ ns} \end{aligned} \quad (5.2)$$

5.5 Bus Arbiter

The bus arbiter is implemented with a 20R8-15 PLD clocked by CLK1. The worst-case margins to this rising edge is calculated as follows.

$$\begin{aligned} \text{CLK1 cycle} &- \text{max 82596SX output} \\ &- \text{min PLD input setup} \\ &+ \text{min CLK2 to CLK1 skew} \\ &= 66 - 32 - 15 + 3.8 = 22.8 \text{ ns} \end{aligned} \quad (5.3)$$

The signal $\overline{BR}$ does not need to meet any setup or hold times because it is internally synchronized by the MC68000. The PLD flip-flop setup time for $\overline{BG}$ is checked next. Because $\overline{BG}$ can go active 0 ns after the falling edge of $\overline{CLK1}$, and because there can be up to 2 ns of skew between CLK1 and $\overline{CLK1}$, it is not completely safe to directly use $\overline{BG}$ in the arbiter PLD. Instead it is run through one of the flip-flops in the PLD to fully synchronize the signal. In the worst case, $\overline{BG}$ can go active about the same time as CLK1 goes high. Because the arbiter's flip-flop will not be clocked until the next rising edge of CLK1, there will be almost 60 ns for the output to reach a valid state.

The outputs of the arbiter, $\overline{LANCYC}$ and $\overline{REFCYC}$, are internally synchronized at their destination, so no output timing analysis is required. An inverter is required for $\overline{LANCYC}$ to create HLDA to the 82596SX. If the PLD has an internal inverter then this will not be required. If not, one of the unused macrocells in the 20R8 can be used to perform the inversion.

5.6 Memory Signal Conversion

The memory signal conversion block is implemented as shown in Section 2.6. A 20R4-15 PLD is used to convert the 82596SX-type control signals to MC68000-type control signals. When the 82596SX does not have the bus all the outputs go to a high-impedance state.

The signals $\overline{UDS}$, $\overline{LDS}$, and A1 are generated by using simple combinatorial decodes of $\overline{BHE}$ and $\overline{BLE}$. The delay will be identical to the PLD propagation delay, which is 15 ns maximum. The signals $\overline{NEWAS}$, $\overline{NEWDS}$, $\overline{NEWDBEN}$, and NEWR/$\overline{W}$ are generated by using the PLD's registered outputs, which are clocked by CLK1. Their states are determined by the state of the 82596SX's signals $\overline{ADS}$, $\overline{RDY}$, and W/$\overline{R}$.

An 82596SX read or write cycle starts with $\overline{ADS}$ going active based on the rising edge of CLK2. $\overline{NEWAS}$ and $\overline{NEWDBEN}$ will go active on the next rising edge of

CLK1. If the cycle is a read cycle $\overline{NEWDS}$ will also go active. If it is a write cycle $\overline{NEWDS}$ will go active one clock later. In general, the signals go inactive based on $\overline{RDY}$ going active. To meet the data hold times $\overline{NEWDBEN}$ stays active one extra clock during a write cycle. NEWR/$\overline{W}$ is simply the inverted and registered W/$\overline{R}$.

The timing of the PLD is checked next. The 82596SX control signals must be valid in time to meet the setup requirements of the PLD's flip-flops. The margin is calculated as follows.

$$\begin{aligned} \text{CLK1 cycle} &+ \text{min CLK2 to CLK1 skew} \\ &- \text{max 82596SX output delay} \\ &- \text{PLD input setup} \\ &= 66 + 3.8 - 36 - 15 = 18.8 \text{ ns} \end{aligned} \quad (5.4)$$

$\overline{NEWAS}$ goes active based on the falling edge of CLK1, which is the same as the rising edge of $\overline{CLK1}$. The PLD clock to output valid delay is 5 to 12 ns maximum. The skew between the clocks will be −2 to +2 ns. This translates to a 3 to 14 ns delay, which is within the MC68000 specifications of 3 to 40 ns.

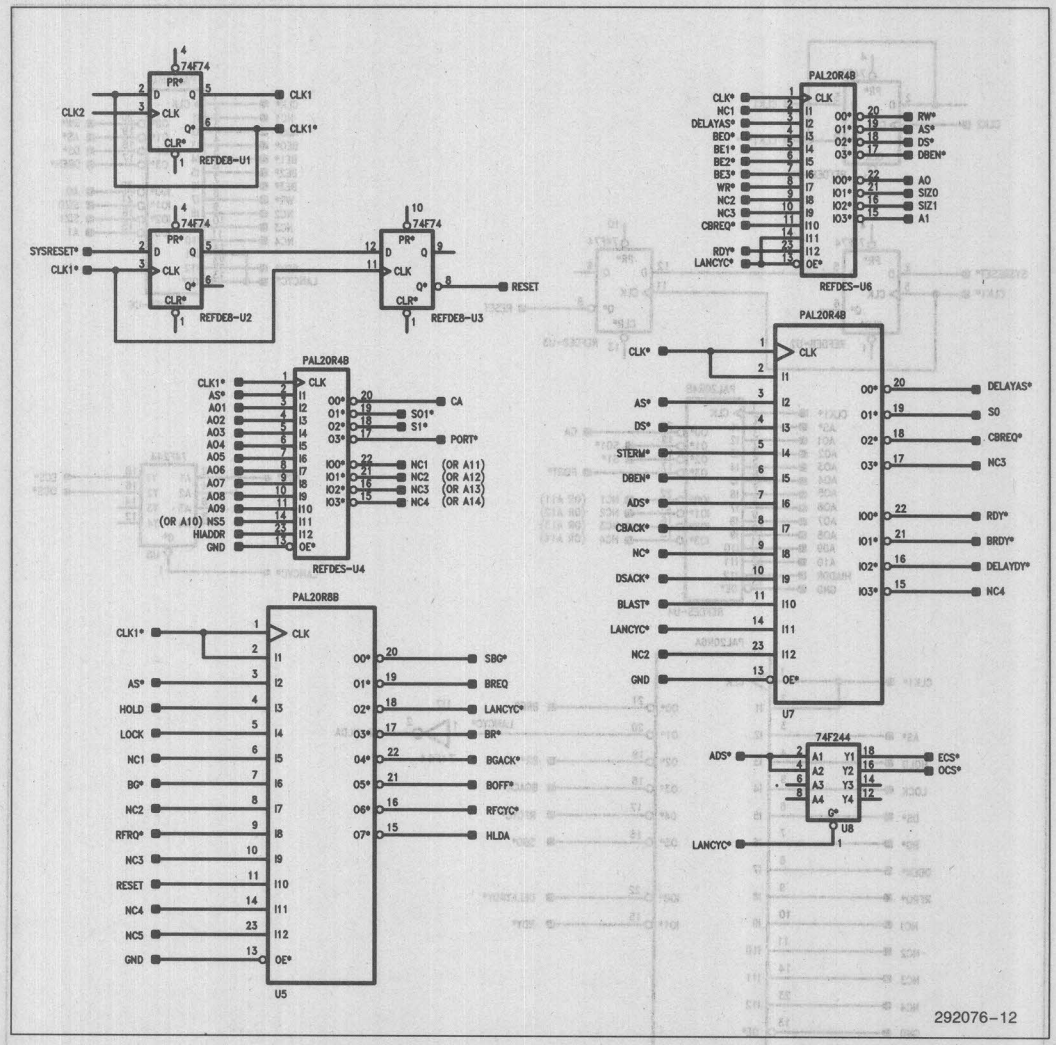
$\overline{ECS}$ and $\overline{OCS}$ are generated by taking $\overline{ADS}$ and running it through a tri-state buffer (74F244) that is enabled by HLDA. When the 82596SX has the bus $\overline{ECS}$ and $\overline{OCS}$ will go active about 8 ns after $\overline{ADS}$.

5.7 Wait State Generator

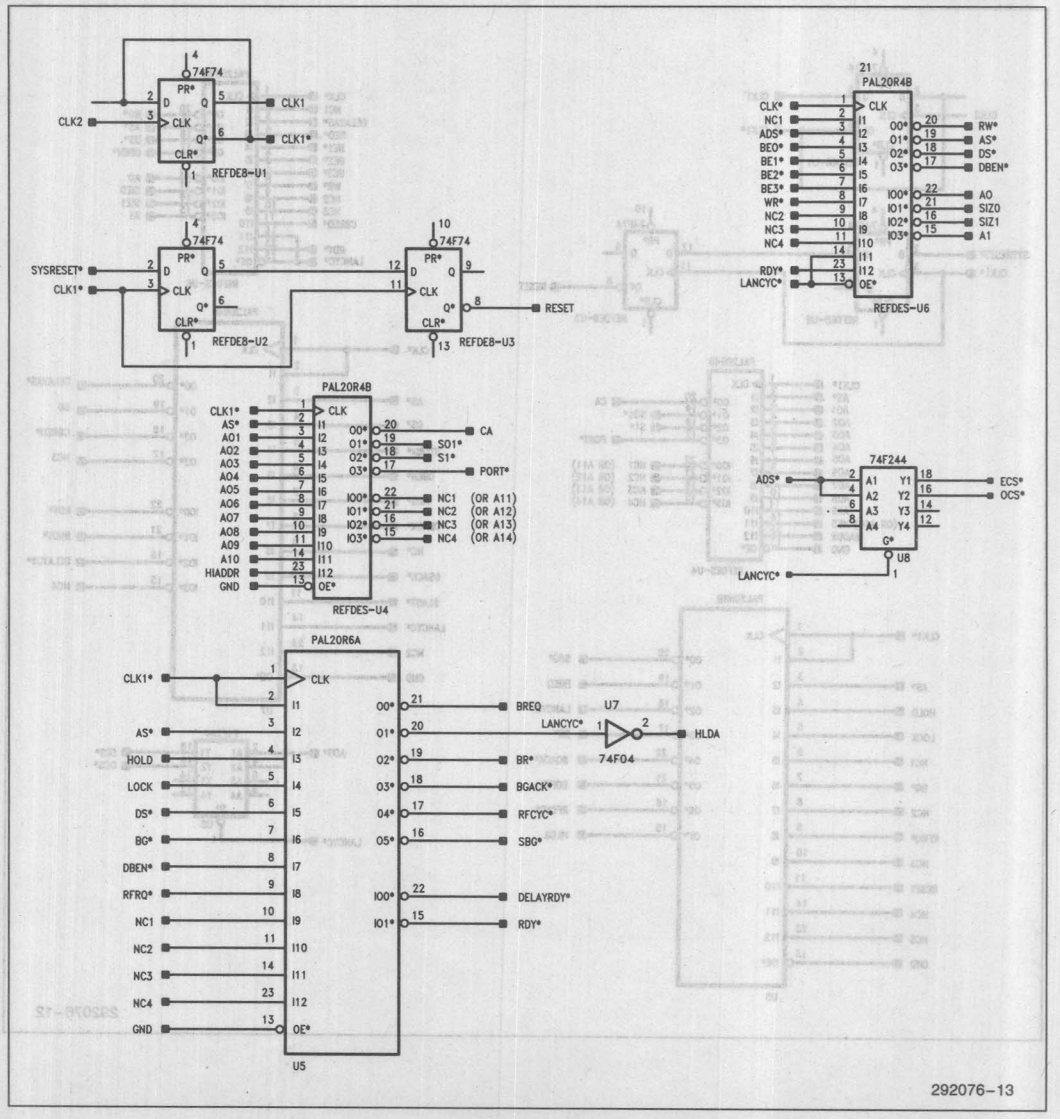
The 82596SX bus cycle is at least two clocks long and begins with $\overline{ADS}$ going active during the first clock and $\overline{RDY}$ active in the last clock. The wait state generation block inserts the proper number of wait states during the bus cycle. For this design it was assumed that the DRAM would allow for zero wait state accesses. If slower DRAMs are used, wait states will need to be inserted in the $\overline{DSACK}$ and $\overline{RDY}$ generation circuits.

The fastest single cycle transfer in this design requires three clocks for the 82596SX. The wait state generator delays the $\overline{RDY}$ signal to the 82596SX. This provides enough time to meet the data setup and hold specifications. The 82596SX requires that data be valid at least 5 ns before the rising edge of its clock. The MC68000 requires only a 1 ns setup to its clock. The system designer will need to guarantee that the 82596SX has at least a 5 ns data setup to this edge, plus 2 ns for the clock skew. If this cannot be met, another wait state will be needed for all 82596 memory read cycles. This can be done by modifying the PLD equations to delay the assertion of $\overline{RDY}$ by one or more clocks.

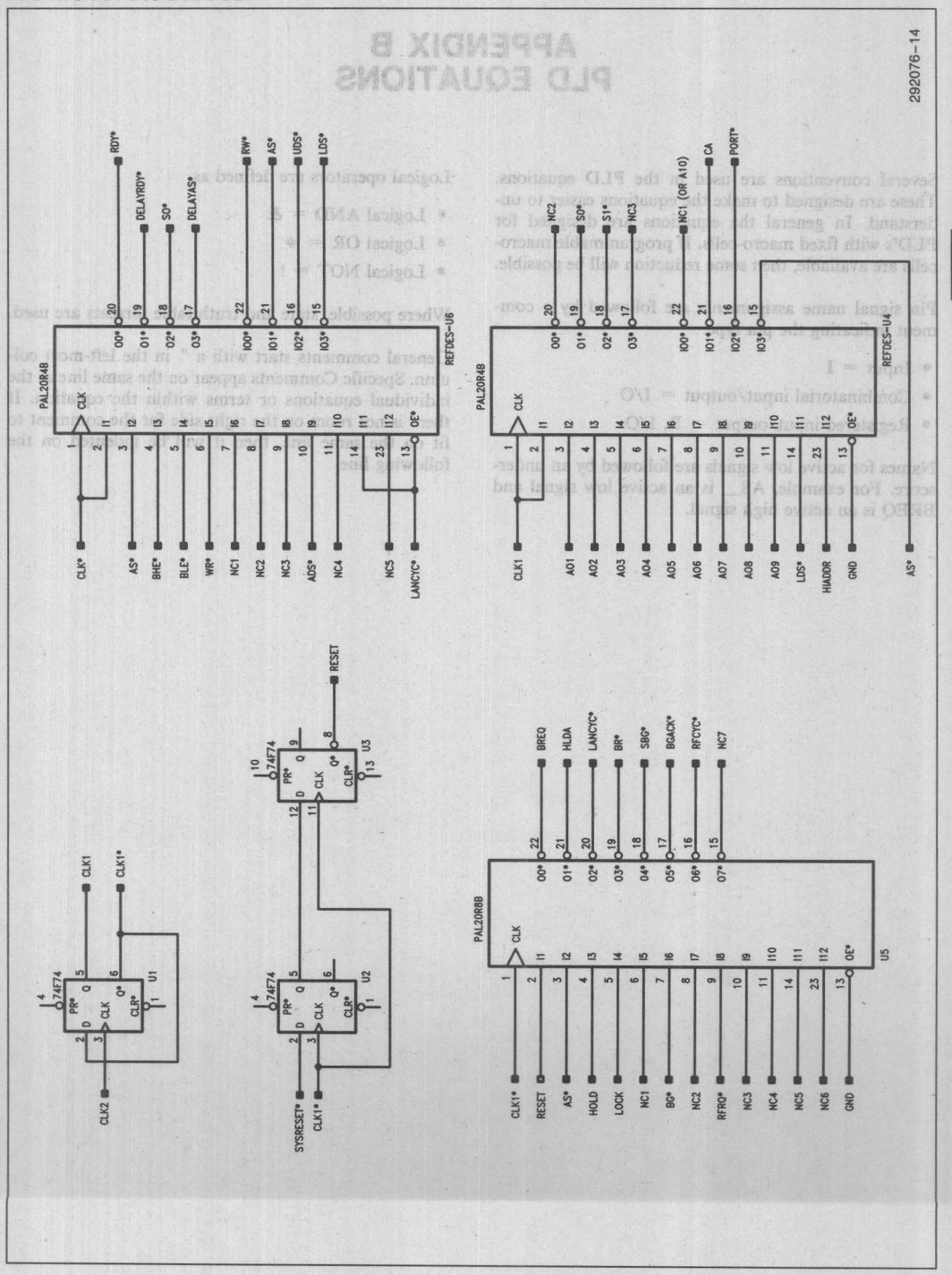
A.1 MC68030/82596CA



A.2 MC68020/82596DX



A.3 MC68000/82596SX



APPENDIX B
PLD EQUATIONS

Several conventions are used in the PLD equations. These are designed to make the equations easier to understand. In general the equations are designed for PLD's with fixed macro-cells. If programmable macro-cells are available, then some reduction will be possible.

Pin signal name assignments are followed by a comment indicating the pin type:

- Input = I
- Combinatorial input/output = I/O
- Registered input/output = R, I/O

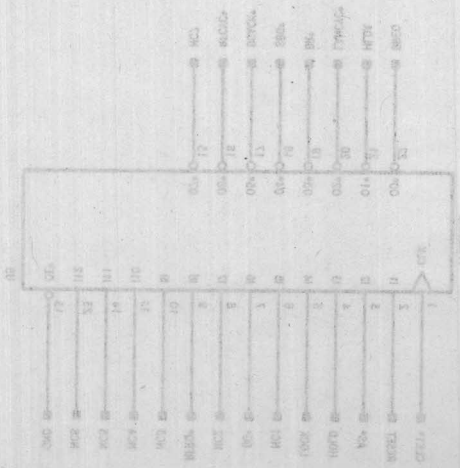
Names for active low signals are followed by an underscore. For example, AS__ is an active low signal and BREQ is an active high signal.

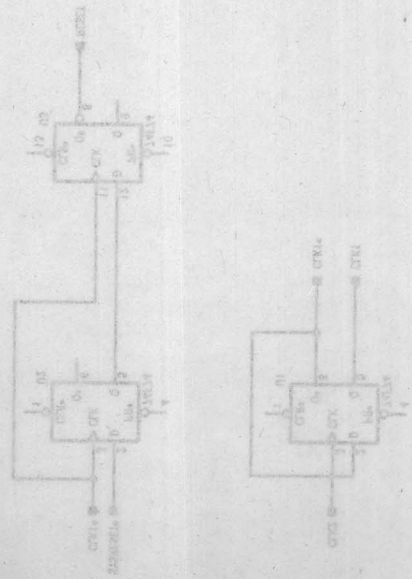
Logical operators are defined as:

- Logical AND = &
- Logical OR = #
- Logical NOT = !

Where possible, state and truth table formats are used.

General comments start with a " in the left-most column. Specific Comments appear on the same line as the individual equations or terms within the equation. If there is not room on the right side for the comment to fit on the same line, then it will be indented on the following line.





B.1 MC68030/82596CA

```
Module ARB  FLAG '-R3';
Title '82596CA Arbitration for Local Bus   Rev. B   02/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R8-10';

" ************************  Descrption   ************************
"
"  FOR:  82596CA / 68030 Interface
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596 requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.
"
"  Requestors are granted the bus by using the inverted CPU clock
"  (CLK1_).  Since CLK1_ is also used in the equations, it must
"  connected with a separate pin with a separate name (CCLK1_).
"
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to the inverted clock
"  (CLK1_).  If this set-up cannot be guaranteed, the request
"  must be synchronized through an external flip-flop, clocked
"  with CLK1_.
"
"  SBG_ is the synchronized 68030 Bus Grant (BG_) signal.

"  HLDA is the inverted LANCYC*.  Due to a lack of P-terms, HLDA
"  will be delayed by 1 clock.  If a PLD with inverter outputs is
"  available, then LANCYC* can be inverted and used directly as
"  HLDA.
"
"  The Bus Request signal, BREQ, and Backoff, BOFF, are used to
"  activiate the Bus Throttle Timers and backoff function.  The
"  equations are included but the outputs are always set
"  inactive.  It is left to the system designer to define input
"  conditions for this signal.
"
"  The two states LAN_OFF and LAN_RF can be used if the external
"  circuitry cannot guarantee that the 82596 will get off the bus
"  in time to do refresh cycles.  If these states are used, a
"  larger PLD will be needed to generate the BGACK to the 68030.
"
"  UNUSED INPUT  PINS                 :  5
"  UNUSED OUTPUT PINS (REGISTERED)    :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL) :  0
"
"*************************************************************
```

292076-35

```
arb Device 'P20R8';

CLK1_     Pin   1; "I"          VCC       Pin 24;
CCLK1_    Pin   2; "I"          NC5       Pin 23; "I"
AS_       Pin   3; "I"          SBG_      Pin 22; "R, I/O"
HOLD      Pin   4; "I"          BREQ      Pin 21; "R, I/O"
LOCK      Pin   5; "I"          LANCYC_   Pin 20; "R, I/O"
NC1       Pin   6; "I"          BR_       Pin 19; "R, I/O"
BG_       Pin   7; "I"          BGACK_    Pin 18; "R, I/O"
NC2       Pin   8; "I"          BOFF_     Pin 17; "R, I/O"
RFRQ_     Pin   9; "I"          RFCYC_    Pin 16; "R, I/O"
NC3       Pin  10; "I"          HLDA      Pin 15; "R, I/O"
RESET     Pin  11; "I"          NC4       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"

MODE    = [BR_, BGACK_, RFCYC_, LANCYC_, BOFF_];

IDLE    = [1,1,1,1,1];
REQ     = [0,1,1,1,1];     " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1,1];     " Refresh request has been granted.
LAN_CYC = [1,0,1,0,1];     " LAN request has been granted.
PRE_CHG = [1,0,1,1,1];     " Required for back-to-back cycles.

"  The following two lines are used only if the 82596 is required to
"  be kicked off the bus.  Most designs will not require these states.

" LAN_OFF = [1,0,1,0,0];    Refresh Request forces 82596 off bus.
" LAN_ON  = [1,0,0,1,0];    Refresh Request returns control to 82596


Equations

BREQ  := 0;                " Bus Throttle conditions will need to be
                           " defined by the system designer.

SBG_ = !BG_ & CCLK1_       " Set synchronized bus grant during high clock
                           "    to phase meet setup to ARB PLD.
     # !SBG_ & !BG_        " Hold until processor bus grant goes away.
     # !SBG_ & !CCLK1_     " Hold through low clock phase to met setup.

HLDA := !LANCYC_;          " Create HLDA from inverted LANCYC_.
```

292076-36

```
MODE  := RESET & IDLE;  " Initialize state machine to IDLE State on reset.

State_Diagram IN arb MODE

state IDLE     : IF (!RFRQ_ # HOLD) THEN REQ
                                    ELSE IDLE;

state REQ      : CASE (!RFRQ_ & !SBG_)                  :RF_CYC;
                      (HOLD & RFRQ_ & !SBG_)            :LAN_CYC;
                      (!((!RFRQ_ & !SBG_)
                       # (HOLD & RFRQ_ & SBG_)))        :REQ;
                 ENDCASE;

state RF_CYC   : CASE (RFRQ_ & !HOLD & SBG_)            :IDLE;
                      (RFRQ_ & HOLD)                    :PRE_CHG;
                      (!RFRQ_)                          :RF_CYC;
                 ENDCASE;

state LAN_CYC  : CASE (!HOLD &  RFRQ_ & !LOCK & SBG_)   :IDLE;
                      (!HOLD & !RFRQ_ & !LOCK)          :PRE_CHG;
"                     ( HOLD & !RFRQ_ & !LOCK)          :LAN_Off;
                      ( HOLD &  RFRQ_)                  :LAN_CYC;
                 ENDCASE;

state PRE_CHG  : CASE ( RFRQ_ & !HOLD & !LOCK & SBG_)   :IDLE;
                      (!RFRQ_ & !LOCK)                  :RF_CYC;
                      ( RFRQ_ &  HOLD)                  :LAN_CYC;
                      ( RFRQ_ & !HOLD & !LOCK & !SBG_)  :PRE_CHG;
                 ENDCASE;

"  state LAN_OFF :   IF (!HOLD) THEN LAN_RF
"                               ELSE LAN_OFF;

"  state LAN_RF  :   IF (RFRQ_) THEN LAN_CYC
"                               ELSE LAN_RF;

" ******************  Revision History  ************************
"
"  Rev. A  01/03/90 - KKP - First Version
"  Rev. B  02/20/90 - KKP - Put BG_ Synchroniziation into PLD.
"
" ***************************************************************

end ARB
```

292076-37

```
"  combinatorial outputs.  They can be
"  inputs.  NC5 is a standard input that is also available as
"  an extra address term.  If even more decode lines are required,
"  then the HIADDR input is for the output of the external
"  decoder.  This decode must be done in less than 16 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ and S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"  UNUSED INPUT  PINS                 :  1
"  UNUSED OUTPUT PINS (REGISTERED)    :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL) :  4
"
" ***************************************************************

caport Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
AS_       Pin   2; "I"          HIADDR    Pin 23; "I"
A01       Pin   3; "I"          NC1       Pin 22; "I/O"
A02       Pin   4; "I"          NC2       Pin 21; "I/O"
A03       Pin   5; "I"          CA        Pin 20; "R,I/O"
A04       Pin   6; "I"          S0_       Pin 19; "R,I/O"
A05       Pin   7; "I"          S1_       Pin 18; "R,I/O"
A06       Pin   8; "I"          PORT_     Pin 17; "R,I/O"
A07       Pin   9; "I"          NC3       Pin 16; "I/O"
A08       Pin  10; "I"          NC4       Pin 15; "I/O"
A09       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"
```

292076-38

```
"Declarations

   X,C    = .X.,.C.;

   ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];   " User defined address.

   CA_ACC    MACRO  { (ADDR == ^h110) & HIADDR & !AS_ };

   PORT_ACC  MACRO  { (ADDR == ^h220) & HIADDR & !AS_ };

MODE      = [CA,PORT_,S0_,S1_];

IDLE      = [ 1,  1,  1 , 1 ];
PORT_SET  = [ 1,  0,  1 , 1 ];  " Set PORT_ to 82596.
PORT_HLD1 = [ 1,  0,  0 , 1 ];  " Hold for one clock state.
PORT_HLD2 = [ 1,  0,  0 , 0 ];  " Hold for a second clock state.
ACCESS_OFF= [ 1,  1,  0 , 0 ];  " Deassert PORT_ and CA.
CA_SET    = [ 0,  1,  1 , 1 ];  " Set CA to 82596.
CA_HLD1   = [ 0,  1,  0 , 1 ];  " Hold for one clock state.
CA_HLD2   = [ 0,  1,  0 , 0 ];  " Hold for a second clock state.


Equations

State_Diagram IN caport MODE

state IDLE       : CASE  (PORT_ACC)                :PORT_SET;
                         (CA_ACC)                  :CA_SET;
                         (!(PORT_ACC # CA_ACC))    :IDLE;
                   ENDCASE;

state PORT_SET   : GOTO PORT_HLD1;

state PORT_HLD1  : GOTO PORT_HLD2;

state PORT_HLD2  : GOTO ACCESS_OFF;

state ACCESS_OFF : IF AS_ THEN IDLE
                      ELSE ACCESS_OFF;

state CA_SET     : GOTO CA_HLD1;

state CA_HLD1    : GOTO CA_HLD2;

state CA_HLD2    : GOTO ACCESS_OFF;


" ***************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ************************************************************

end CAPORT
```

292076-39

```
Module CNVRT  FLAG '-R3';
Title '82596CA Signal Conversion          Rev. A    01/03/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-10';

" ************************   Descrption   ************************
"
"  FOR:  82596CA / 68030 Interface
"
"  This PLD converts the 82596 signals to 68030 type signals.
"
"  DELAYAS_ is generated in the RDY PLD to delay AS_ until it is
"  known whether a multiple or burst transfer is to take place.
"
"  A PLD 20R4 was used in this example, requiring seperate
"  output enables (LANCYC_ and LANCYC2_, connected external to
"  the PLD) for the registered and latched outputs.
"
"  NEWRW_, NEWAS_, NEWDS_, and NEWDBEN_ are registered outputs.
"  NEWSIZ0, NEWSIZ1, NEWA0, and NEWA1 are combinatorial outputs.
"  All of these signals will be enabled when the 82596 has
"  control of the local bus, otherwise they will be tri-stated.
"
"  The combinatorial outputs are generated using a truth table.
"  For completeness, default settings are included for the
"  impossible BE# input combinations.
"
"  UNUSED INPUT  PINS                  :  3
"  UNUSED OUTPUT PINS (REGISTERED)     :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  :  0
"
" ***************************************************************

cnvrt Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
NC1       Pin   2; "I"          RDY_      Pin 23; "I"
DELAYAS_  Pin   3; "I"          NEWA0     Pin 22; "I/O"
BE0_      Pin   4; "I"          NEWSIZ0   Pin 21; "I/O"
BE1_      Pin   5; "I"          NEWRW_    Pin 20; "R,I/O"
BE2_      Pin   6; "I"          NEWAS_    Pin 19; "R,I/O"
BE3_      Pin   7; "I"          NEWDS_    Pin 18; "R,I/O"
WR_       Pin   8; "I"          NEWDBEN_  Pin 17; "R,I/O"
NC2       Pin   9; "I"          NEWSIZ1   Pin 16; "I/O"
NC3       Pin  10; "I"          NEWA1     Pin 15; "I/O"
NEWCBRQ_  Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND       Pin  12;              LANCYC_   Pin 13; "I"
```

292076-40

```
Equations

ENABLE NEWAS_    = !LANCYC_;
ENABLE NEWDS_    = !LANCYC_;
ENABLE NEWDBEN_  = !LANCYC_;
ENABLE NEWRW_    = !LANCYC_;
ENABLE NEWSIZ0   = !LANCYC2_;
ENABLE NEWSIZ1   = !LANCYC2_;
ENABLE NEWA0     = !LANCYC2_;
ENABLE NEWA1     = !LANCYC2_;

!NEWAS_   :=

    !DELAYAS_                  " Start after BLAST_ valid.
  # !NEWAS_ & !NEWCBRQ_;       " Hold through multiple/burst transfer.

!NEWDS_   :=

    !WR_ & !DELAYAS_           " Start DS_ with AS_ during read cycle.
  #  WR_ & !NEWDBEN_ & RDY_    " Delay DS_ by 1 clock during a write cycle.
  # !NEWDS_ & !NEWCBRQ_;       " Hold until clock following RDY_ set.

!NEWDBEN_ :=

    !DELAYAS_                    " Enable data transceivers as soon as 82596
                                 "    begins its cycle.
  # !WR_ & !NEWDBEN_ & RDY_      " Hold as long as AS_ during read.
  #  WR_ & !NEWDBEN_ & !NEWAS_;  " Longer data hold during a write.

!NEWRW_   :=  WR_;               "invert WR to match processor

" The following truth table converts the byte enable signals from
" the 82596 into the 68030 SIZ signals and address lines A0 and
" A1.

Truth_Table

( [BE3_,BE2_,BE1_,BE0_,LANCYC2_] -> [NEWSIZ1,NEWSIZ0,NEWA1,NEWA0] )

  [ X , X , X , X ,   1   ] -> [  1  ,  1  ,  1  ,  1  ];
  [ 1 , 1 , 1 , 1 ,   0   ] -> [  1  ,  1  ,  1  ,  1  ];
  [ 1 , 1 , 1 , 0 ,   0   ] -> [  0  ,  1  ,  1  ,  1  ];
  [ 1 , 1 , 0 , 0 ,   0   ] -> [  1  ,  0  ,  1  ,  0  ];
  [ 1 , 0 , 0 , 0 ,   0   ] -> [  1  ,  1  ,  0  ,  1  ];
  [ 0 , 0 , 0 , 0 ,   0   ] -> [  0  ,  0  ,  0  ,  0  ];
  [ 1 , 1 , 0 , 1 ,   0   ] -> [  0  ,  1  ,  1  ,  0  ];
  [ 1 , 0 , 0 , 1 ,   0   ] -> [  1  ,  0  ,  0  ,  1  ];
  [ 0 , 0 , 0 , 1 ,   0   ] -> [  1  ,  1  ,  0  ,  0  ];
  [ 1 , 0 , 1 , 1 ,   0   ] -> [  0  ,  1  ,  0  ,  1  ];
  [ 0 , 0 , 1 , 1 ,   0   ] -> [  1  ,  0  ,  0  ,  0  ];
  [ 0 , 1 , 1 , 1 ,   0   ] -> [  0  ,  1  ,  0  ,  0  ];
```

292076-41

```
"  The following BE# input combinations are illegal and will
"  result in erroneous data transfers.

  [ X  , 0  , 1  , 0  ,   1    ] -> [   1   ,    1  , 1  , 1  ];
  [ 0  , 1  , 0  , X  ,   1    ] -> [   1   ,    1  , 1  , 1  ];
  [ 0  , 1  , 1  , 0  ,   1    ] -> [   1   ,    1  , 1  , 1  ];

" ****************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ************************************************************

end CNVRT
```

292076-42

```
Module RDY  FLAG '-R3';
Title '82596CA Ready and Burst Cycle Logic  Rev. A     01/03/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-10';

" ************************  Descrption  ************************
"
"  This PLD generates the RDY_ and BRDY_ signals to the 82596.
"  It also generates the Burst Request (CBREQ_) signal to the
"  memory controller.  It uses the 68030 signals Address Strobe
"  (AS_), Data Strobe (DS_), Data Bus Enable (DBEN_), Data
"  Acknowledge (DSACK_) and Synchronous Termination (STERM_).  It
"  also uses Cache Burst Acknowledge (CBACK_) from the memory
"  controller and Burst Last (BLAST_) from the 82596.  The
"  DELAYAS_ signal is used to delay the generation of AS_ to the
"  memory controller in order to determine whether a burst transfer
"  is about to take place.  Because CLK1_ is needed for both the
"  flip-flop registers and in the combinatorial equations, it
"  is connected to both pins 1 and 2.  Two separate names are
"  required in the equations (CLK1_ and CCLK1_).
"
"  The first three burst data transfers between the 82596 and the
"  memory will be acknowledged with the BRDY_ signal.  The last
"  (fourth) burst data transfer cycle will be acknowledged with a
"  RDY_.
"
"  This PLD must be 10 ns or faster to meet BRDY set-up to CLK1_
"  on 82596.
"
"  The output DELAYRDY_ is only used inside this PLD to generate
"  a delay for the RDY_ signal to the 82596.
"
"  UNUSED INPUT  PINS                 :  2
"  UNUSED OUTPUT PINS (REGISTERED)    :  1
"  UNUSED OUTPUT PINS (COMBINATORIAL) :  1
"
" ************************************************************

rdy Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
CCLK1_    Pin   2; "I"          NC2       Pin 23; "I"
AS_       Pin   3; "I"          RDY_      Pin 22; "I/O"
DS_       Pin   4; "I"          BRDY_     Pin 21; "I/O"
STERM_    Pin   5; "I"          DELAYAS_  Pin 20; "R,I/O"
DBEN_     Pin   6; "I"          S0        Pin 19; "R,I/O"
ADS_      Pin   7; "I"          NEWCBREQ_ Pin 18; "R,I/O"
CBACK_    Pin   8; "I"          NC3       Pin 17; "R,I/O"
NC1       Pin   9; "I"          DELAYRDY_ Pin 16; "I/O"
DSACK_    Pin  10; "I"          NC4       Pin 15; "I/O"
BLAST_    Pin  11; "I"          LANCYC_   Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"
```

292076-43

```
"Declarations

MODE      = [DELAYAS_,S0,NEWCBREQ_];

IDLE      = [1,1,1];
BLST_WT   = [0,1,1];  " Wait for BLAST to determine if burst data transfer.
NO_BURST  = [1,0,1];  " BLAST_ active so no burst transfer.
BRST_CYC  = [0,0,1];  " BLAST_ not active so multiple or burst transfer.
STERM_1   = [0,0,0];  " Wait for acknowledge.
STERM_2   = [0,1,0];  " Wait for acknowledge.
STERM_3   = [1,1,0];  " Wait for acknowledge.

Equations

!RDY_    = !BLAST_ & !DELAYRDY_ & CCLK1_ & !AS_ & CBACK_
             " Return RDY_ whenever BLAST_ asserted.
         # !STERM_ & NEWCBREQ_
             " Fourth burst transfer or synchronous transfer.
         # !RDY_ & !DBEN_;
             " Hold RDY_ until data requirement met.

!BRDY_   = !STERM_ & !NEWCBREQ_ & !CBACK_
             " Assert BRDY_ during burst cycles
         # !BRDY_ & !CCLK1_;
             " Hold so recognized on rising edge of CLK1_ to 82596.

!DELAYRDY_ =

     !DS_ & !LANCYC_ & !CCLK1_    " Delay RDY for data setup.
   # !DELAYRDY_ & !AS_;           " Hold until end of data cycle.


State_Diagram IN rdy MODE

state IDLE     : IF !ADS_ THEN BLST_WT
                          ELSE IDLE;

state BLST_WT  : IF !BLAST_ THEN NO_BURST
                            ELSE BRST_CYC;

state NO_BURST : GOTO IDLE;

state BRST_CYC : CASE (!BRDY_ & !CBACK_)            : STERM_1;
                      ((!BRDY_ & CBACK_) # !BLAST_) : IDLE;
                      (BRDY_ & BLAST_)              : BRST_CYC;
                 ENDCASE;

state STERM_1  : CASE (!BRDY_ & !CBACK_)            : STERM_2;
                      ((!BRDY_ & CBACK_) # !BLAST_) : IDLE;
                      (BRDY_ & BLAST_)              : STERM_1;
                 ENDCASE;
```

292076-44

```
state STERM_2  : CASE (!BRDY_ & !CBACK_)              : STERM_3;
                      ((!BRDY_ & CBACK_) # !BLAST_)   : IDLE;
                      (BRDY_ & BLAST_)                : STERM_2;
                 ENDCASE;

state STERM_3  : IF (!BRDY_ # !BLAST_) THEN IDLE
                                       ELSE STERM_3;

" ***************  Revision History  ***************************
"
"  Rev. A    1/20/90 - KKP - First Version.
"
" ********************************************************************

end RDY
```

292076-45

B.2 MC68020/82596DX

```
Module ARB  FLAG '-R3';
Title '82596DX Arbitration for Local Bus    Rev. A     01/12/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R6-10';

" ************************   Descrption   ************************
"
"  FOR:  82596DX / 68020 Interface
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596DX requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.  The RDY_ acknowledge signal to
"  the 82596DX is also generated in this PLD.  The signal
"  DELAYRDY_ is an embedded signal used only in this PLD to
"  generate RDY_.
"
"  Requestors are granted the bus by using the inverted CPU clock
"  (CLK1).  Because is required for both the registered and
"  combinatorial terms, CLK1 is connected to both the clock and a
"  combinatorial input.  The combinatorial term is called CCLK1.
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to CLK1.  If this
"  set-up cannot be guaranteed, the request must be synchronized
"  through an external flip-flop, clocked with CLK1.
"
"  The SBG_ signal is the synchronized 68020 Bus Grant (BG_)
"  signal.
"
"  Because the 82596 uses and active-high HOLD, LANCYC_ is
"  inverted with an external 74F04.
"
"  The equations and marco-cells are allocated for the Bus
"  Request (BREQ) signal, which is used to activiate the 82596DX
"  Bus Throttle Timers.  In these equations it is set inactive.
"  It is left to the system designer to define input conditions
"  for this signal.
"
"  UNUSED INPUT  PINS                  : 4
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 0
"
" ************************************************************
```

292076-46

```
arb Device 'P20R6';

CLK1      Pin   1; "I"          VCC       Pin 24;
CCLK1     Pin   2; "I"          NC3       Pin 23; "I"
AS_       Pin   3; "I"          DELAYRDY_ Pin 22; "I/O"
HOLD      Pin   4; "I"          BREQ      Pin 21; "R,I/O"
LOCK      Pin   5; "I"          LANCYC_   Pin 20; "R,I/O"
DS_       Pin   6; "I"          BR_       Pin 19; "R,I/O"
BG_       Pin   7; "I"          BGACK_    Pin 18; "R,I/O"
DBEN_     Pin   8; "I"          RFCYC_    Pin 17; "R,I/O"
RFRQ_     Pin   9; "I"          SBG_      Pin 16; "R,I/O"
NC1       Pin  10; "I"          RDY_      Pin 15; "I/O"
NC2       Pin  11; "I"          NC4       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"

MODE    = [BR_,BGACK_,RFCYC_,LANCYC_];

IDLE    = [1,1,1,1];
REQ     = [0,1,1,1];     " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1];     " Refresh request has been granted.
LAN_CYC = [1,0,1,0];     " LAN request has been granted.
PRE_CHG = [1,0,1,1];     " Required for back-to-back cycles.

Equations

BREQ  := 0;              " Bus Throttle conditions will need
                         " to be defined by the system designer.

!SBG_  = !BG_ & CCLK1    " Set during high phase of ARB clock to
                         "   meet setup time into PLD.
       # !SBG_ & !BG_     " Hold with processor bus grant.
       # !SBG_ & !CCLK1;  " Hold through low phase of clock to meet setup time.

!DELAYRDY_ =

     !DS_ & !LANCYC_ & !CCLK1    " Delay RDY_ for data set-up.
   # !DELAYRDY_ & !AS_;          " Hold until end of data cycle.

!RDY_  =

     !DELAYRDY_ & CCLK1 & !AS_   " Return RDY_ after delay while
                                 "   data cycle still in progress.
   # !RDY_ & !DBEN_;             " Hold until end of data cycle.
```

292076-47

1

```
State_Diagram IN arb MODE

state IDLE     : IF (!RFRQ_ # HOLD) THEN REQ
                                    ELSE IDLE;

state REQ      : CASE (!RFRQ_ & !SBG_)          :RF_CYC;
                      ( HOLD & RFRQ_)           :LAN_CYC;
                      (!((!RFRQ_ & !SBG_)
                       # (HOLD & RFRQ_ & SBG_)))  :REQ;
                 ENDCASE;

state RF_CYC   : CASE (RFRQ_ & !HOLD)           :IDLE;
                      (RFRQ_ & HOLD)            :PRE_CHG;
                      (!RFRQ_)                  :RF_CYC;
                 ENDCASE;

state LAN_CYC  : CASE (!HOLD & RFRQ_ & !LOCK)     :IDLE;
                      (!HOLD & !RFRQ_ & !LOCK)    :PRE_CHG;
                      (HOLD)                      :LAN_CYC;
                 ENDCASE;

state PRE_CHG  : CASE (RFRQ_ & !HOLD & !LOCK)     :IDLE;
                      (RFRQ_ & HOLD)              :LAN_CYC;
                      (!RFRQ_ & !LOCK)            :RF_CYC;
                      (!SBG_ & RFRQ_ & !HOLD & !LOCK)  :PRE_CHG;
                 ENDCASE;

" ******************  Revision History  ************************
"
"  Rev. A  01/12/90 - KKP - First Version
"  Rev. B  02/20/90 - KKP - Moved BG_ synchronization into PLD.
"
" ***************************************************************

end ARB
```

292076-48

```
Module CAPORT  FLAG '-R3';
Title '82596DX Channel Attention and Port   Rev. A   01/12/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';
" ************************   Descrption   ************************
"
"  FOR:   82596DX / 68020 Interface
"
"  This PLD decodes the 68020 address lines and generates the
"  Channel Attention and PORT_ to the 82596DX.  The choice of
"  address is left to the system designer.
"
"  Nine address decode lines are available.  They could be
"  connected to A31-A23.  NC1, NC2, NC3, and NC4 are
"  combinatorial outputs.  They can be used as extra address
"  inputs.  NC5 is a standard input that is also available as
"  an extra address term.  If even more decode lines are required,
"  then the HIADDR input is for the output of the external
"  decoder.  This decode must be done in less than 22 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ AND S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"
"  UNUSED INPUT  PINS                  :  1
"  UNUSED OUTPUT PINS (REGISTERED)     :  0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  :  4
""
" ***************************************************************

caport Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
AS_       Pin   2; "I"          HIADDR    Pin 23; "I"
A01       Pin   3; "I"          NC1       Pin 22; "I/O"
A02       Pin   4; "I"          NC2       Pin 21; "I/O"
A03       Pin   5; "I"          CA        Pin 20; "R,I/O"
A04       Pin   6; "I"          S0_       Pin 19; "R,I/O"
A05       Pin   7; "I"          S1_       Pin 18; "R,I/O"
A06       Pin   8; "I"          PORT_     Pin 17; "R,I/O"
A07       Pin   9; "I"          NC3       Pin 16; "I/O"
A08       Pin  10; "I"          NC4       Pin 15; "I/O"
A09       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE        Pin 13; "I"
```

292076-49

```
" Declarations

   X,C    = .X.,.C.;

   ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];   " User defined address.

   CA_ACC    MACRO  ( (ADDR == ^h110) & HIADDR & !AS_ );

   PORT_ACC  MACRO  ( (ADDR == ^h220) & HIADDR & !AS_ );
```

292076-50

```
MODE       = [CA,PORT_,S0_,S1_];

IDLE       = [ 1,  1,  1 , 1 ];
PORT_SET   = [ 1,  0,  1 , 1 ];  " Set PORT_ to 82596DX.
PORT_HLD1  = [ 1,  0,  0 , 1 ];  " Hold for one clock state.
PORT_HLD2  = [ 1,  0,  0 , 0 ];  " Hold for a second clock state.
ACCESS_OFF = [ 1,  1,  0 , 0 ];  " Deassert PORT_ and CA.
CA_SET     = [ 0,  1,  1 , 1 ];  " Set CA to 82596DX.
CA_HLD1    = [ 0,  1,  0 , 1 ];  " Hold for one clock state.
CA_HLD2    = [ 0,  1,  0 , 0 ];  " Hold for a second clock state.

Equations

State_Diagram IN caport MODE

state IDLE       : CASE   (PORT_ACC)          :PORT_SET;
                          (CA_ACC)            :CA_SET;
                          (!(PORT_ACC # CA_ACC))  :IDLE;
                   ENDCASE;

state PORT_SET   : GOTO PORT_HLD1;

state PORT_HLD1  : GOTO PORT_HLD2;

state PORT_HLD2  : GOTO ACCESS_OFF;

state ACCESS_OFF : IF AS_ THEN IDLE
                          ELSE ACCESS_OFF;

state CA_SET     : GOTO CA_HLD1;

state CA_HLD1    : GOTO CA_HLD2;

state CA_HLD2    : GOTO ACCESS_OFF;

" ***************  Revision History  ***************************
"
"  Rev. A    1/3/90 - KKP - First Version.
"
" ************************************************************

end CAPORT
```

292076-51

1

```
Module CNVRT  FLAG '-R3';
Title '82596DX Signal Conversion          Rev. A     1/12/90
      DOCTOR DESIGN, San Diego, CA
      PLD20R4-10';

" ************************  Descrption  ************************
"
"  FOR:   82596DX / 68020 Interface
"
"  This PLD converts the 82596DX signals to 68020 type signals.
"
"  These signals will be enabled when the 82596DX has control of
"  the local bus (LANCYC_ is low), otherwise they will be
"  tri-stated.
"
"  A PLD 20R4 was used in this example, requiring seperate
"  enables, LANCYC_ and LANCYC2_, which are the same signal
"  external to the PLD.  If the PLD does not require separate
"  output enables for registered and latched outputs then this is
"  not required.
"
"  NEWRW_, NEWAS_, NEWDS_, and NEWDBEN_ are registered outputs.
"  NEWSIZ0, NEWSIZ1, NEWA0, and NEWA1 are combinatorial outputs.
"
"  The combinatorial outputs are generated using a truth table.
"  For completeness, default settings are included for the
"  impossible BE# input combinations.
"
"  UNUSED INPUT  PINS                  : 4
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 0
"
" ***************************************************************

cnvrt Device 'P20R4';

CLK1_     Pin   1; "I"          VCC       Pin 24;
NC1       Pin   2; "I"          RDY_      Pin 23; "I"
ADS_      Pin   3; "I"          NEWA0     Pin 22; "I/O"
BE0_      Pin   4; "I"          NEWSIZ0   Pin 21; "I/O"
BE1_      Pin   5; "I"          NEWRW_    Pin 20; "R,I/O"
BE2_      Pin   6; "I"          NEWAS_    Pin 19; "R,I/O"
BE3_      Pin   7; "I"          NEWDS_    Pin 18; "R,I/O"
WR_       Pin   8; "I"          NEWDBEN_  Pin 17; "R,I/O"
NC2       Pin   9; "I"          NEWSIZ1   Pin 16; "I/O"
NC3       Pin  10; "I"          NEWA1     Pin 15; "I/O"
NC4       Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND       Pin  12;              LANCYC_   Pin 13; "I"


"Declarations

     X = .X.;
```

292076-52

```
Equations

ENABLE NEWAS_    = !LANCYC_;
ENABLE NEWDS_    = !LANCYC_;
ENABLE NEWDBEN_  = !LANCYC_;
ENABLE NEWRW_    = !LANCYC_;
ENABLE NEWSIZ0   = !LANCYC2_;
ENABLE NEWSIZ1   = !LANCYC2_;
ENABLE NEWA0     = !LANCYC2_;
ENABLE NEWA1     = !LANCYC2_;

!NEWAS_ :=

     !ADS_                  " Start AS_ during 68020 clock low cycle.
   # !NEWAS_ & RDY_;        " Hold until clock following RDY_ set.

!NEWDS_ :=

     !WR_ & !ADS_               " Start DS_ with AS_ during read.
   #  WR_ & !NEWDBEN_ & RDY_   " Delay DS_ by 1 clock during write.
   # !NEWDS_ & RDY_;           " Hold until clock following RDY_ set.

!NEWDBEN_ :=

     !ADS_                          " Enable data transceivers as soon
                                    " as 82596DX begins its cycle.
   # !WR_ & !NEWDBEN_ & RDY_        " Hold as long as AS_ during read.
   #  WR_ & !NEWDBEN_ & !NEWAS_;    " Longer data hold during a write.

!NEWRW_   :=  WR_;                  " Invert W/R_ to match processor.


" The following truth table converts the byte enable signals from
" the 82596DX into the 68020 SIZE signals and address lines A0
"  and A1.


Truth_Table

( [BE3_,BE2_,BE1_,BE0_,LANCYC2_] -> [NEWSIZ1,NEWSIZ0,NEWA1,NEWA0] )

  [ X , X , X , X ,   1   ] -> [  1  ,   1  ,  1  ,  1  ];
  [ 1 , 1 , 1 , 1 ,   0   ] -> [  1  ,   1  ,  1  ,  1  ];
  [ 1 , 1 , 1 , 0 ,   0   ] -> [  0  ,   1  ,  1  ,  1  ];
  [ 1 , 1 , 0 , 0 ,   0   ] -> [  1  ,   0  ,  1  ,  0  ];
  [ 1 , 0 , 0 , 0 ,   0   ] -> [  1  ,   1  ,  0  ,  1  ];
  [ 0 , 0 , 0 , 0 ,   0   ] -> [  0  ,   0  ,  0  ,  0  ];
  [ 1 , 1 , 0 , 1 ,   0   ] -> [  0  ,   1  ,  1  ,  0  ];
  [ 1 , 0 , 0 , 1 ,   0   ] -> [  1  ,   0  ,  0  ,  1  ];
  [ 0 , 0 , 0 , 1 ,   0   ] -> [  1  ,   1  ,  0  ,  0  ];
  [ 1 , 0 , 1 , 1 ,   0   ] -> [  0  ,   1  ,  0  ,  1  ];
  [ 0 , 0 , 1 , 1 ,   0   ] -> [  1  ,   0  ,  0  ,  0  ];
  [ 0 , 1 , 1 , 1 ,   0   ] -> [  0  ,   1  ,  0  ,  0  ];
```

292076-53

```
"  The following BE# input combinations are illegal and will
"  result in erroneous data transfers.

  [ X  , 0  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 0  , X  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];
  [ 0  , 1  , 1  , 0  ,   1    ] -> [   1   ,    1  ,  1  ,  1  ];

" ***************** Revision History ****************************
"
"  Rev. A    01/12/90 - KKP - First Version.
"
" ***************************************************************

end CNVRT
```

292076-54

B.3 MC68000/82596SX

```
Module ARB  FLAG '-R3';
Title '82596SX Arbitration for Local Bus   Rev. A   01/20/90
      DOCTOR DESIGN, San Diego, CA
      PLD20R8-15';

" ************************   Descrption   ************************
"
"  This PLD arbitrates between the CPU, the LAN Controller and
"  the Refresh requestor for the local bus.  Refresh requests are
"  given highest priority, and the 82596SX requests given second
"  highest priority.  The CPU normally controls the local bus
"  when no requests are pending.
"
"  Requestors are granted the bus by using the 82596SX clock,
"  CLK1.
"
"  The refresh request (RFRQ_) is assumed to be an active low
"  signal having the required 12 ns set-up to the clock (CLK1).
"  If this set-up cannot be guaranteed, the request must be
"  synchronized through an external flip-flop, clocked with CLK1.
"
"  The SBG_ signal is the synchronized 68000 Bus Grant (BG__)
"  signal.  It is be synchronized internally using a flip-flop
"  clocked with CLK1.  Because the 82596SX uses an active high
"  HLDA, LANCYC_ is inverted using one of the macro-cells.
"
"  The equations and macro-cells are allocated for the Bus
"  Request signal, which is used to activiate the Bus Throttle
"  Timers.  It is left to the system designer to define input
"  conditions for this signal.
"
"  UNUSED INPUT  PINS                  : 6
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 1
"
" ************************************************************

arb Device 'P20R8';

CLK1      Pin   1; "I"          VCC       Pin 24;
RESET     Pin   2; "I"          NC6       Pin 23; "I"
AS_       Pin   3; "I"          BREQ      Pin 22; "R,I/O"
HOLD      Pin   4; "I"          HLDA      Pin 21; "R,I/O"
LOCK      Pin   5; "I"          LANCYC_   Pin 20; "R,I/O"
NC1       Pin   6; "I"          BR_       Pin 19; "R,I/O"
BG_       Pin   7; "I"          SBG_      Pin 18; "R,I/O"
NC2       Pin   8; "I"          BGACK_    Pin 17; "R,I/O"
RFRQ_     Pin   9; "I"          RFCYC_    Pin 16; "R,I/O"
NC3       Pin  10; "I"          NC7       Pin 15; "R,I/O"
NC4       Pin  11; "I"          NC5       Pin 14; "I"
GND       Pin  12;              OE_       Pin 13; "I"
```

292076-55

```
MODE    = [BR_,BGACK_,RFCYC_,LANCYC_];

IDLE    = [1,1,1,1];
REQ     = [0,1,1,1];    " Generic request to CPU for local bus.
RF_CYC  = [1,0,0,1];    " Refresh request has been granted.
LAN_CYC = [1,0,1,0];    " LAN request has been granted.
PRE_CHG = [1,0,1,1];    " Required for back-to-back cycles.


Equations

BREQ  := 0;             " Bus Throttle conditions will need to be
                        "  defined by the system designer.

HLDA  := !LANCYC_;      " 82596SX requires active-high HLDA.

SBG_  := BG_;           " Synchronized Bus Grant.

MODE  := RESET & IDLE;  " Initialize state machine to IDLE State on reset

State_Diagram IN arb MODE

state IDLE     : IF (!RFRQ_ # HOLD) THEN REQ
                                    ELSE IDLE;

state REQ      : CASE (!RFRQ_ & !SBG_)                    :RF_CYC;
                      (HOLD & RFRQ_)                      :LAN_CYC;
                      (!((!RFRQ_ & !SBG_)
                        # (HOLD & RFRQ_ & SBG_)))         :REQ;
                 ENDCASE;

state RF_CYC   : CASE (RFRQ_ & !HOLD)                     :IDLE;
                      (RFRQ_ & HOLD)                      :PRE_CHG;
                      (!RFRQ_)                            :RF_CYC;
                 ENDCASE;

state LAN_CYC  : CASE (!HOLD & RFRQ_ & !LOCK)             :IDLE;
                      (!HOLD & !RFRQ_ & !LOCK)            :PRE_CHG;
                      (HOLD)                              :LAN_CYC;
                 ENDCASE;

state PRE_CHG  : CASE (RFRQ_ & !HOLD & !LOCK)             :IDLE;
                      (RFRQ_ & HOLD)                      :LAN_CYC;
                      (!RFRQ_ & !LOCK)                    :RF_CYC;
                      (!SBG_ & RFRQ_ & !HOLD & !LOCK)     :PRE_CHG;
                 ENDCASE;

" ******************  Revision History  ************************
"
"  Rev. A  1/20/90 - KKP - First Version
"
" ***************************************************************

end ARB
```

292076-56

```
Module CAPORT  FLAG '-R3';
Title '82596SX Channel Attention and Port   Rev. A   1/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';
" ************************  Descrption   ************************
"
"  FOR:  82596SX / 68000 Interface
"
"  This PLD decodes the 68000 address lines and generates Channel
"  Attention and PORT_ to the 82596.  The choice of address is
"  left to the system designer.
"
"  Nine address decode lines are available.  They could be
"  connected to A23-A14.  NC1 is a combinatorial outputs and
". could be used as extra address input.  If even more decode
"  lines are required, then the HIADDR input is for the output
"  of the external decoder.  This decode must be done in less
"  than 60 ns.
"
"  In the line ADDR = [A09,A08,....], the values for A09-A01
"  should be set high or low (inverted) for the desired range.
"  The decode values for CA_ACC and PORT_ACC (110 and 220) are
"  arbitrary and can be modified as needed.
"
"  S0_ AND S1_ are state bits used for generating wait states for
"  PORT_ assertion.
"
"  UNUSED INPUT  PINS                  : 0
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 1
"  UNUSED OUTPUT PINS (REGISTERED)     : 2
"
"*****************************************************************************

caport Device 'P20R4';

CLK1     Pin   1; "I"          VCC      Pin 24;
CCLK1    Pin   2; "I"          HIADDR   Pin 23; "I"
A01      Pin   3; "I"          NC1      Pin 22; "I/O"
A02      Pin   4; "I"          CA       Pin 21; "I/O"
A03      Pin   5; "I"          NC2      Pin 20; "R,I/O"
A04      Pin   6; "I"          S0_      Pin 19; "R,I/O"
A05      Pin   7; "I"          S1_      Pin 18; "R,I/O"
A06      Pin   8; "I"          NC3      Pin 17; "R,I/O"
A07      Pin   9; "I"          PORT_    Pin 16; "I/O"
A08      Pin  10; "I"          AS_      Pin 15; "I/O"
A09      Pin  11; "I"          LDS_     Pin 14; "I"
GND      Pin  12;              OE_      Pin 13; "I"
```

292076-57

```
"Declarations

  X,C    = .X.,.C.;

  ADDR   = [A09,A08,A07,A06,A05,A04,A03,A02,A01,X,X,X];     " User defined address.

  CA_ACC    MACRO  ( (ADDR == ^h110) & HIADDR & !AS_ );

  PORT_ACC  MACRO  ( (ADDR == ^h220) & HIADDR & !AS_ );

MODE     = [S0_,S1_];

IDLE     = [ 1,  1 ];
STR_CNT  = [ 0 , 1 ];  " PORT_ or CA has been sent to 82596.
CNT_1    = [ 0,  0 ];  " Hold for one clock state.
CNT_2    = [ 1,  0 ];  " Hold for a second clock state.


Equations

!CA    =

    !LDS_ & !AS_ & CCLK1 & CA_ACC  " Start when data valid on bus.
  # !CA & !(S0_ & !S1_)             " Hold for at least 2 clocks.
  # !CA & CCLK1;                    " Guarantee CA hold time to 82596.

!PORT_ = !LDS_ & !AS_ & CCLK1 & PORT_ACC      " Start when data valid on bus.
       # !PORT_ & !(S0_ & !S1_)               " Hold for at least 2 clocks.
       # !PORT_ & CCLK1;                      " Guarantee PORT_ hold time to 82596.


State_Diagram IN caport MODE

state IDLE       : IF (!PORT_ # CA) THEN STR_CNT
                                    ELSE IDLE;

state STR_CNT    : GOTO CNT_1;

state CNT_1      : GOTO CNT_2;

state CNT_2      : IF AS_ THEN IDLE
                          ELSE CNT_2;

" ***************  Revision History  ***************************
"
"  Rev. A    1/20/90 - KKP - First Version.
"
" ************************************************************

end CAPORT
```

292076-58

```
Module RDY  FLAG '-R3';
Title '82596SX Ready and Signal Conversion  Rev. A   01/20/90
       DOCTOR DESIGN, San Diego, CA
       PLD20R4-15';


" ************************   Descrption   ************************
"
"  This PLD generates the RDY_ signal to the 82596SX.  It also
"  converts the 82596SX signals BHE_, BLE_, ADS_ and WR_ to the
"  68000 equivalents, UDS, LDS, AS_, and RW_ and mimics their
"  timing to the memory controller.
"
"  The output DELAYRDY_ is only used inside this PLD to generate
"  a delay for the RDY_ signal to the 82596SX.
"
"  A 20R4 was used for this example requiring a separate input
"  for the combinatorial enable.
"
"  UNUSED INPUT  PINS                  : 5
"  UNUSED OUTPUT PINS (COMBINATORIAL)  : 0
"  UNUSED OUTPUT PINS (REGISTERED)     : 0
"
"*************************************************************

rdy Device 'P20R4';

CLK1     Pin   1; "I"          VCC       Pin 24;
CCLK1    Pin   2; "I"          NC5       Pin 23; "I"
AS_      Pin   3; "I"          NEWRW_    Pin 22; "I/O"
BHE_     Pin   4; "I"          NEWAS_    Pin 21; "I/O"
BLE_     Pin   5; "I"          RDY_      Pin 20; "R,I/O"
WR_      Pin   6; "I"          DELAYRDY_ Pin 19; "R,I/O"
NC1      Pin   7; "I"          S0_       Pin 18; "R,I/O"
NC2      Pin   8; "I"          DELAYAS_  Pin 17; "R,I/O"
NC3      Pin   9; "I"          NEWUDS_   Pin 16; "I/O"
ADS_     Pin  10; "I"          NEWLDS_   Pin 15; "I/O"
NC4      Pin  11; "I"          LANCYC2_  Pin 14; "I"
GND      Pin  12;              LANCYC_   Pin 13; "I"

MODE     = [DELAYAS_,DELAYRDY_,RDY_,S0_];

IDLE     = [1,1,1,1];
STR_AS   = [0,1,1,1];  " Delay AS_ until clock phase of 68000 S2.
DLY_RDY  = [0,0,1,1];  " Delay RDY_ by 1 82596 clock state.
DLY_DS   = [0,0,1,0];  " Delay UDS_,LDS_ during write cycle.
STR_RDY  = [0,0,0,1];  " Initiate RDY_ for 68000 type 0 wait cycle.
```

292076-59

```
   # !NEWUDS_ & WR_ & RDY;

!NEWUDS_    =

     !WR_ & !DELAYAS_ & !BLE_ & !CCLK1 & !AS_      " Start LDS_ with AS_ on read.
   #  WR_ & !S0_ & !BLE_ & !CCLK1 & !AS_           " Delay LDS_ on a write.
   # !NEWUDS_ & !WR_ & !DELAYAS_                   " Hold thru data cycle.
   # !NEWUDS_ & WR_ & RDY;

!NEWAS_     = !DELAYAS_ & !CCLK1
            # !NEWAS_ & !WR_ & !DELAYAS_
            # !NEWAS_ & WR_ & RDY;


State_Diagram IN rdy MODE

state IDLE     : IF !ADS_ THEN STR_AS
                          ELSE IDLE;

state STR_AS   : IF !WR_  THEN DLY_RDY
                          ELSE DLY_DS;

state DLY_DS   : GOTO DLY_RDY;

state DLY_RDY  : GOTO STR_RDY;

state STR_RDY  : GOTO IDLE;

" ***************  Revision History  ************************
"
"  Rev. A    1/20/90 - KKP - First Version.
" ******************************************************************

end RDY
```

292076-60

APPENDIX C
TIMING DIAGRAMS

The following section includes the timing diagram for each specific design. A summary of the timing specifications is also included.

C.1 MC68030/82596CA

- Block Diagram
- MC68030 and 82596CA Clock Synchronization
- MC68030 and 82596CA CA and PORT Access
- MC68030 Local Arbitration (1 page)
- 82596CA Memory Access (2 pages)
- Timing Summary

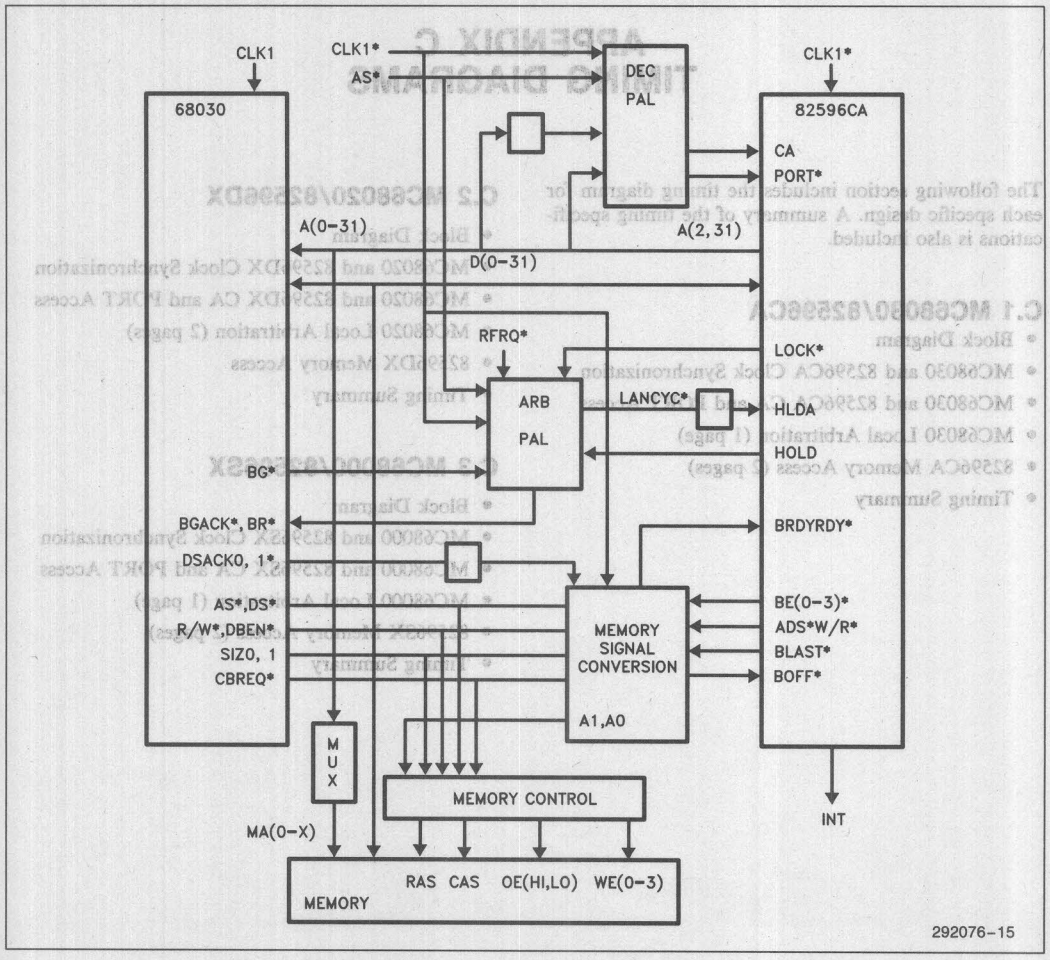
C.2 MC68020/82596DX

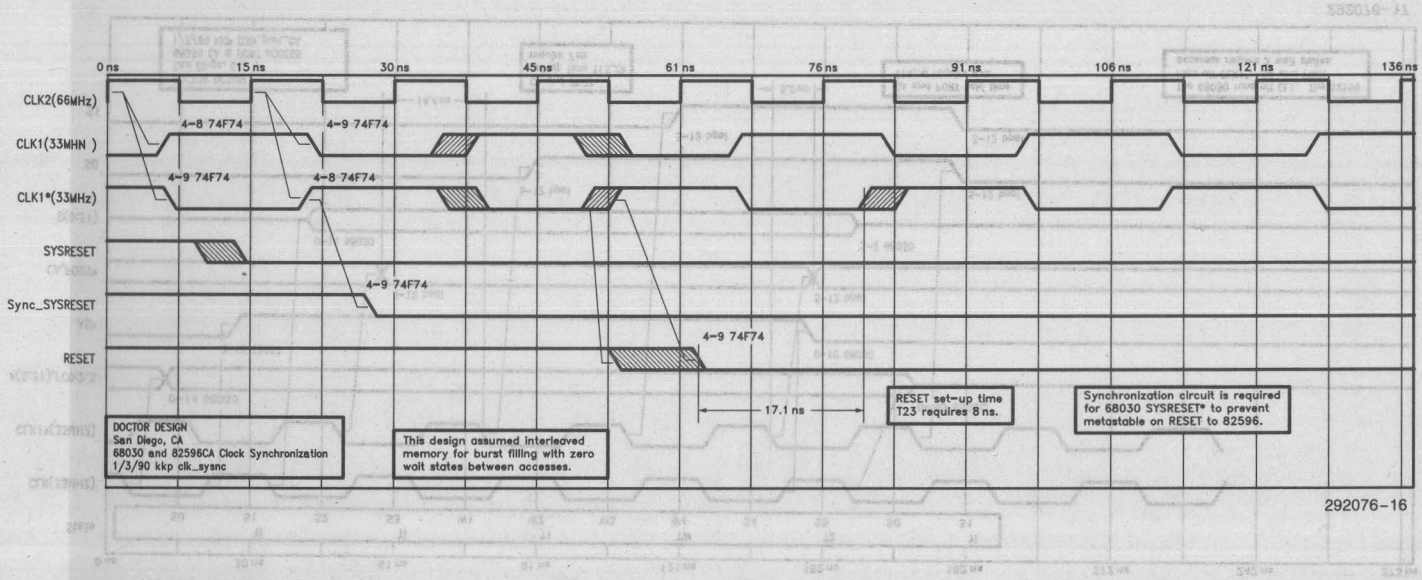
- Block Diagram
- MC68020 and 82596DX Clock Synchronization
- MC68020 and 82596DX CA and PORT Access
- MC68020 Local Arbitration (2 pages)
- 82596DX Memory Access
- Timing Summary

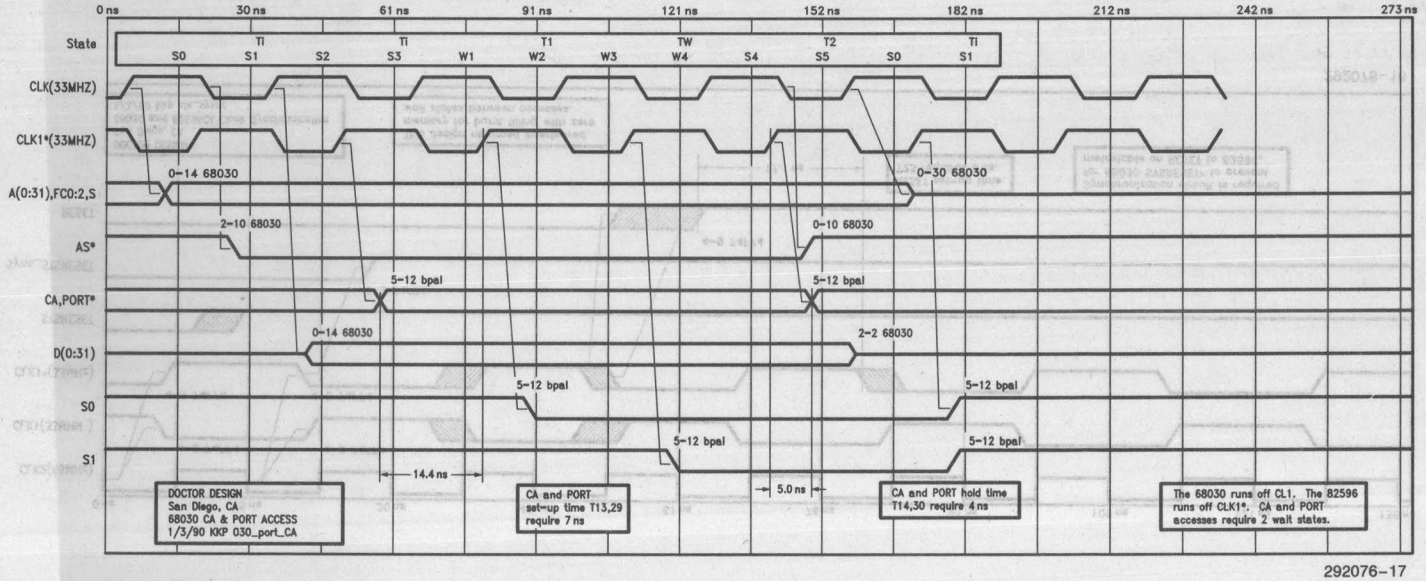
C.3 MC68000/82596SX

- Block Diagram
- MC68000 and 82596SX Clock Synchronization
- MC68000 and 82596SX CA and PORT Access
- MC68000 Local Arbitration (1 page)
- 82596SX Memory Access (2 pages)
- Timing Summary

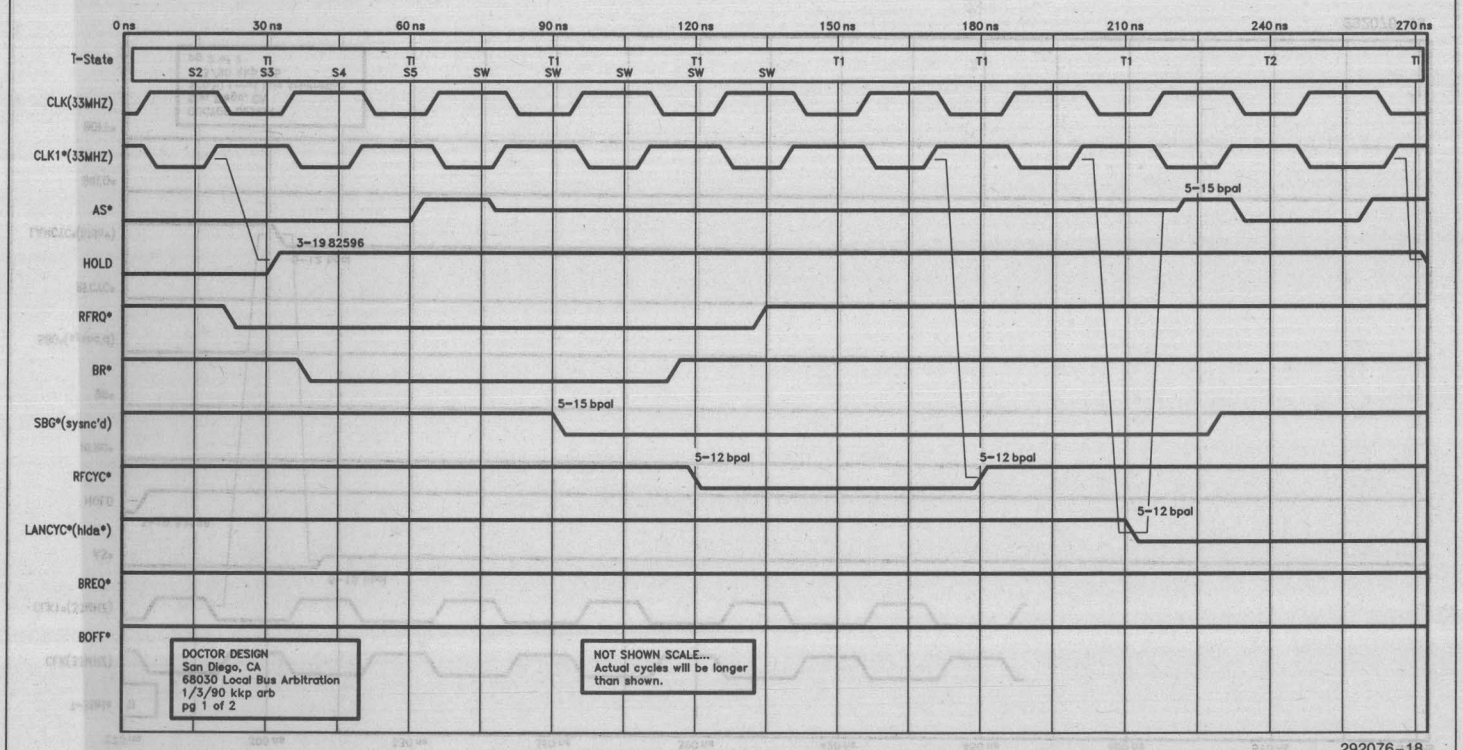
INTERFACE BETWEEN 68030 AND 82596CA LAN CONTROLLER



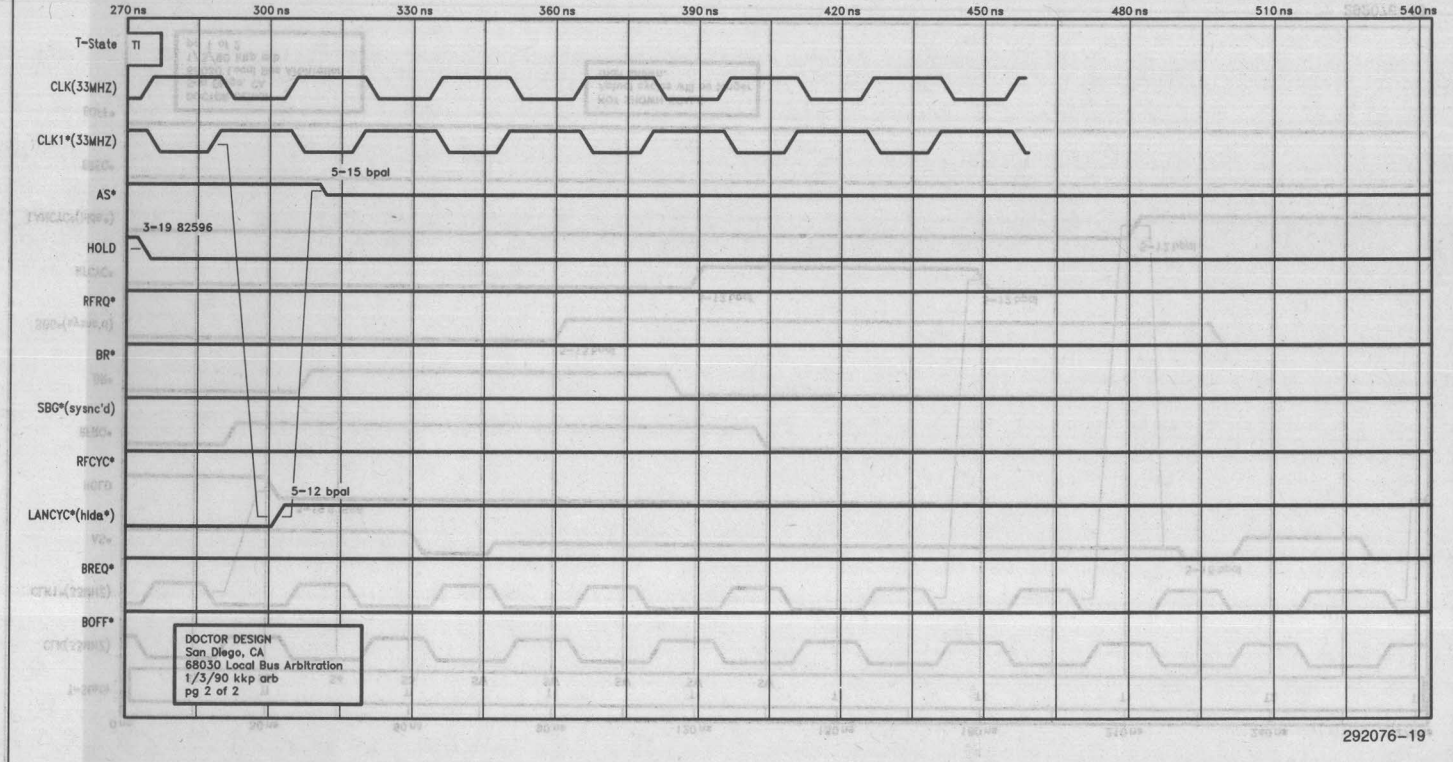

0 ns
15 ns
30 ns
45 ns
61 ns
76 ns
91 ns
106 ns
121 ns
136 ns
CLK2(66MHz)
CLK1(33MHN)
CLK1*(33MHz)
SYSRESET
Sync_SYSRESET
RESET
4-8 74F74
4-9 74F74
4-9 74F74
4-8 74F74
4-9 74F74
4-9 74F74
17.1 ns
RESET set-up time T23 requires 8 ns.
Synchronization circuit is required for 68030 SYSRESET* to prevent metastable on RESET to 82596.
DOCTOR DESIGN
San Diego, CA
68030 and 82596CA Clock Synchronization
1/3/90 kkp clk_sysnc
This design assumed interleaved memory for burst filling with zero wait states between accesses.
292076-16

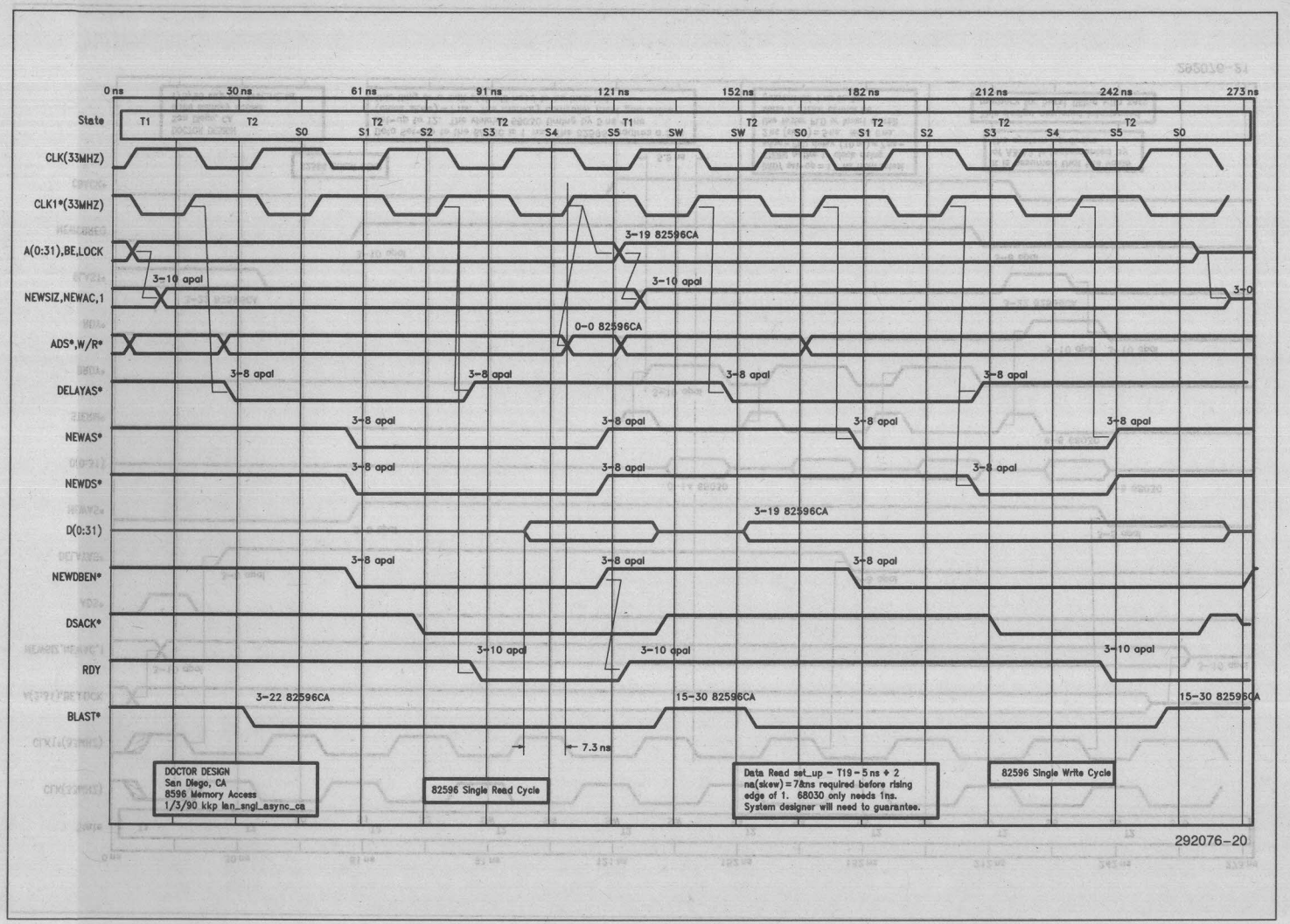

0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
TI
T1
TW
T2
S0
S1
S2
S3
W1
W2
W3
W4
S4
S5
CLK(33MHZ)
CLK1*(33MHZ)
A(0:31),FC0:2,S
AS*
CA,PORT*
D(0:31)
S0
S1
0-14 68030
2-10 68030
0-30 68030
0-10 68030
5-12 bpal
2-2 68030
14.4 ns
5.0 ns
DOCTOR DESIGN
San Diego, CA
68030 CA & PORT ACCESS
1/3/90 KKP 030_port_CA
CA and PORT
set-up time T13,29
require 7 ns
CA and PORT hold time
T14,30 require 4 ns
The 68030 runs off CL1. The 82596
runs off CLK1*. CA and PORT
accesses require 2 wait states.

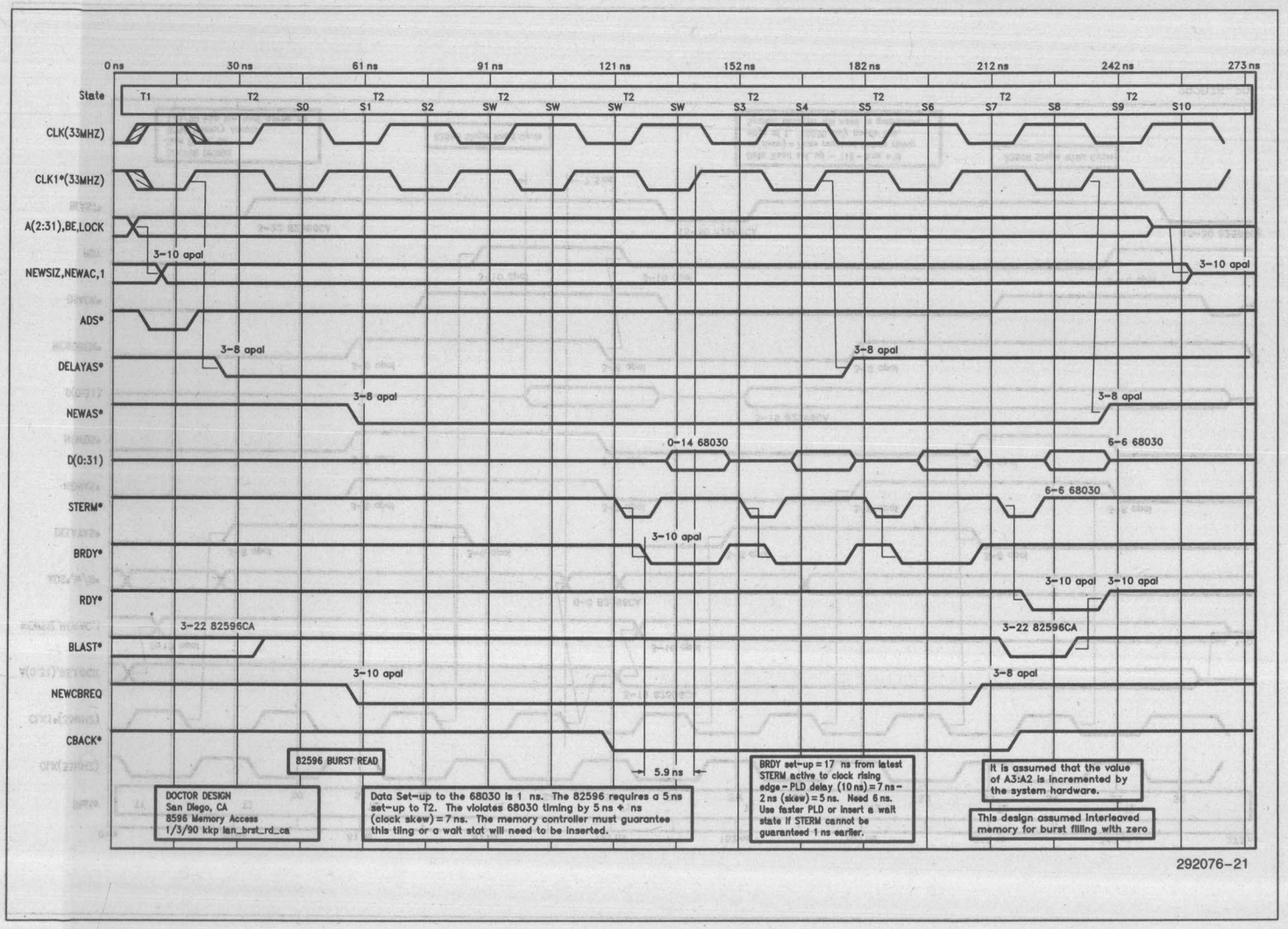
292076-17


0 ns
30 ns
60 ns
90 ns
120 ns
150 ns
180 ns
210 ns
240 ns
270 ns
T-State
S2
T1
S3
S4
T1
S5
SW
T1
SW
SW
T1
SW
SW
T1
T1
T1
T2
T1
CLK(33MHZ)
CLK1*(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBG*(sysnc'd)
RFCYC*
LANCYC*(hlda*)
BREQ*
BOFF*
3-19 82596
5-15 bpal
5-15 bpal
5-12 bpal
5-12 bpal
5-12 bpal
DOCTOR DESIGN
San Diego, CA
68030 Local Bus Arbitration
1/3/90 kkp arb
pg 1 of 2
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.

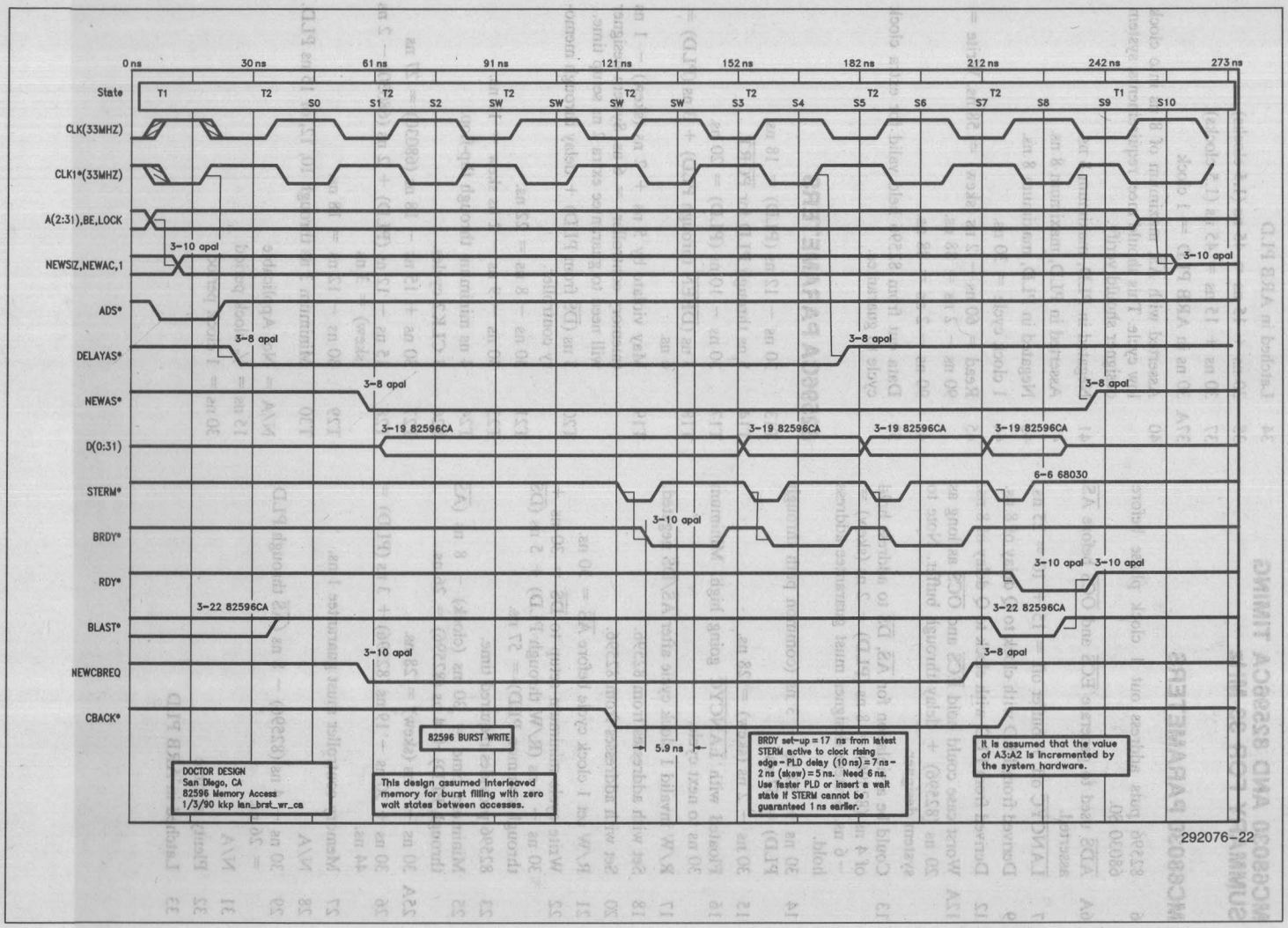
292076-18


270 ns
300 ns
330 ns
360 ns
390 ns
420 ns
450 ns
480 ns
510 ns
540 ns
T-State
TI
CLK(33MHZ)
CLK1*(33MHZ)
AS*
5-15 bpal
HOLD
3-19 82596
RFRQ*
BR*
SBG*(sysnc'd)
RFCYC*
5-12 bpal
LANCYC*(hlda*)
BREQ*
BOFF*
DOCTOR DESIGN
San Diego, CA
68030 Local Bus Arbitration
1/3/90 kkp arb
pg 2 of 2
292076-19


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
T1 T2 S0 S1 S2 S3 S4 S5 SW SW S0 S1 S2 S3 S4 S5 S0
CLK(33MHZ)
CLK1*(33MHZ)
A(0:31),BE,LOCK
NEWSIZ,NEWAC,1
ADS*,W/R*
DELAYAS*
NEWAS*
NEWDS*
D(0:31)
NEWDBEN*
DSACK*
RDY
BLAST*
3-19 82596CA
3-10 apal
0-0 82596CA
3-8 apal
3-22 82596CA
15-30 82596CA
7.3 ns
DOCTOR DESIGN
San Diego, CA
8596 Memory Access
1/3/90 kkp lan_sngl_async_ca
82596 Single Read Cycle
Data Read set_up - T19 - 5 ns + 2 ns(skew) = 7&ns required before rising edge of 1. 68030 only needs 1ns. System designer will need to guarantee.
82596 Single Write Cycle
292076-20


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
T1 T2 S0 T2 S1 T2 S2 T2 SW SW SW SW T2 S3 T2 S4 T2 S5 S6 T2 S7 T2 S8 T2 S9 S10
CLK(33MHZ)
CLK1*(33MHZ)
A(2:31),BE,LOCK
NEWSIZ,NEWAC,1
3-10 apal
3-10 apal
ADS*
DELAYAS*
3-8 apal
3-8 apal
NEWAS*
3-8 apal
3-8 apal
D(0:31)
0-14 68030
6-6 68030
STERM*
6-6 68030
BRDY*
3-10 apal
RDY*
3-10 apal
3-10 apal
BLAST*
3-22 82596CA
3-22 82596CA
NEWCBREQ
3-10 apal
3-8 apal
CBACK*
82596 BURST READ
5.9 ns
DOCTOR DESIGN
San Diego, CA
8596 Memory Access
1/3/90 kkp lan_brst_rd_cs
Data Set-up to the 68030 is 1 ns. The 82596 requires a 5 ns set-up to T2. The violates 68030 timing by 5 ns + ns (clock skew) = 7 ns. The memory controller must guarantee this tiing or a wait stat will need to be inserted.
BRDY set-up = 17 ns from latest STERM active to clock rising edge - PLD delay (10 ns) = 7 ns - 2 ns (skew) = 5 ns. Need 6 ns. Use faster PLD or insert a wait state if STERM cannot be guaranteed 1 ns earlier.
It is assumed that the value of A3:A2 is incremented by the system hardware.
This design assumed interleaved memory for burst filling with zero

292076-21


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
T1 T2 S0 T2 S1 T2 S2 T2 SW SW T2 SW SW T2 S3 T2 S4 T2 S5 T2 S6 T2 S7 T2 S8 T1 S9 S10
CLK(33MHZ)
CLK1*(33MHZ)
A(2:31),BE,LOCK
NEWSIZ,NEWAC,1
ADS*
DELAYAS*
NEWAS*
D(0:31)
STERM*
BRDY*
RDY*
BLAST*
NEWCBREQ
CBACK*
3-10 apal
3-8 apal
3-19 82596CA
6-6 68030
3-22 82596CA
82596 BURST WRITE
5.9 ns
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/3/90 kkp lan_brst_wr_ca
This design assumed interleaved memory for burst filling with zero wait states between accesses.
BRDY set-up = 17 ns from latest STERM active to clock rising edge - PLD delay (10 ns) = 7 ns - 2 ns (skew) = 5 ns. Need 6 ns. Use faster PLD or insert a wait state if STERM cannot be guaranteed 1 ns earlier.
It is assumed that the value of A3:A2 is incremented by the system hardware.
292076-22

MC68030 AND 82596CA TIMING SUMMARY FOR 33 MHz

MC68030 PARAMETERS

6 82596 puts address out 1 clock phase before 68030 S0.

6A $\overline{ADS}$ used to generate $\overline{ECS}$ and $\overline{OCS}$ before $\overline{AS}$ asserted.

7 $\overline{LANCYC}$ off + buffer off = 15 + 10 = 25 ns.

9 Derived from PLD with clock to Q delay of 8 ns.

12 Derived from PLD with clock to Q delay of 8 ns.

12A Worst case could hold $\overline{ECS}$ and $\overline{OCS}$ as long as 20 ns (82596) + delay through buffer. Note to system designer.

13 Could be a violation for $\overline{AS}$, $\overline{DS}$ to address hold of 4 ns (82596) = 8 ns (PLD) − 2 ns (skew) = −6 ns. System designer must guarantee address hold.

14 30 ns + 30 ns − 3 ns (common path through PLD) = 57 ns.

15 30 ns − 2 ns (skew) = 28 ns.

16 Floated with $\overline{LANCYC}$ going high. Minimum 30 ns to next cycle.

17 R/W invalid 1 clock cycle after $\overline{AS}$/$\overline{DS}$ negated.

18 Set with addresses from 82596.

20 Set with addresses from 82596.

21 R/W set 1 clock cycle before $\overline{AS}$ = 30 ns.

22 Write cycle minimum setup to $\overline{DS}$ = 30 ns + 30 ns − 8 ns (R/W through PLD) + 5 ns ($\overline{DS}$ through common PLD) = 57 ns.

23 82596 provides required time.

25 Minimum time = 30 ns (clock) − 8 ns ($\overline{AS}$ through PLD) + 4 ns (82596) = 26 ns.

25A 30 ns − 2 ns (skew) = 28 ns.

26 30 ns + 30 ns − 19 ns (82596) + 3 ns (PLD) = 44 ns.

27 Memory controller must guarantee 1 ns.

28 N/A

29 30 ns + 4 ns (82596) − 8 ns ($\overline{AS}$ through PLD) = 26 ns.

31 N/A

32 Plenty of time

33 Latched in ARB PLD

34 Latched in ARB PLD

35 30 ns + 15 ns = 45 ns (1.5 clocks).

37 30 ns + 15 ns = 45 ns (1.5 clocks).

37A 30 ns in ARB PLD = 1 clock

40 Asserted with $\overline{AS}$, maximum of 8 ns into clock low cycle. This should meet requirements, system designer should verify.

41 Negated in PLD, maximum 8 ns.

42 Asserted in PLD, maximum 8 ns.

43 Negated in PLD, maximum 8 ns.

44 1 clock cycle = 30 ns.

45 Read = 60 ns − 2 ns (skew) = 58 ns. Write = 90 ns − 2 ns = 88 ns.

46 90 ns − 2 ns = 88 ns.

53 Data out from 82596 held valid for extra clock cycle to guarantee.

82596CA PARAMETERS

T13 30 ns − 12 ns (PLD) = 18 ns.

T14 5 ns through PLD for $\overline{PORT}$.

T17 30 ns − 10 ns (PLD) = 20 ns.

T18 3 ns ($\overline{DBEN}$ through PLD) + 3 ns (PLD) = 6 ns.

T19 May violate by 5 ns + 2 ns (skew) − 1 ns (memory controller) = 6 ns. System designer will need to guarantee extra 2 ns setup time.

T20 3 ns ($\overline{DS}$ from PLD) + delay through memory controller.

T21 30 ns − 8 ns = 22 ns.

T23 30 ns − 9 ns − 2 ns skew = 19 ns.

T24 4 ns minimum through flip-flop.

T26 3 CLK2 cycles.

T27 30 ns + 15 ns − 18 ns (68030) = 27 ns

T28 15 ns − 12 ns (PLD) + 2 ns (68030) − 2 ns (skew) = 3 ns

T29 30 ns − 12 ns = 18 ns.

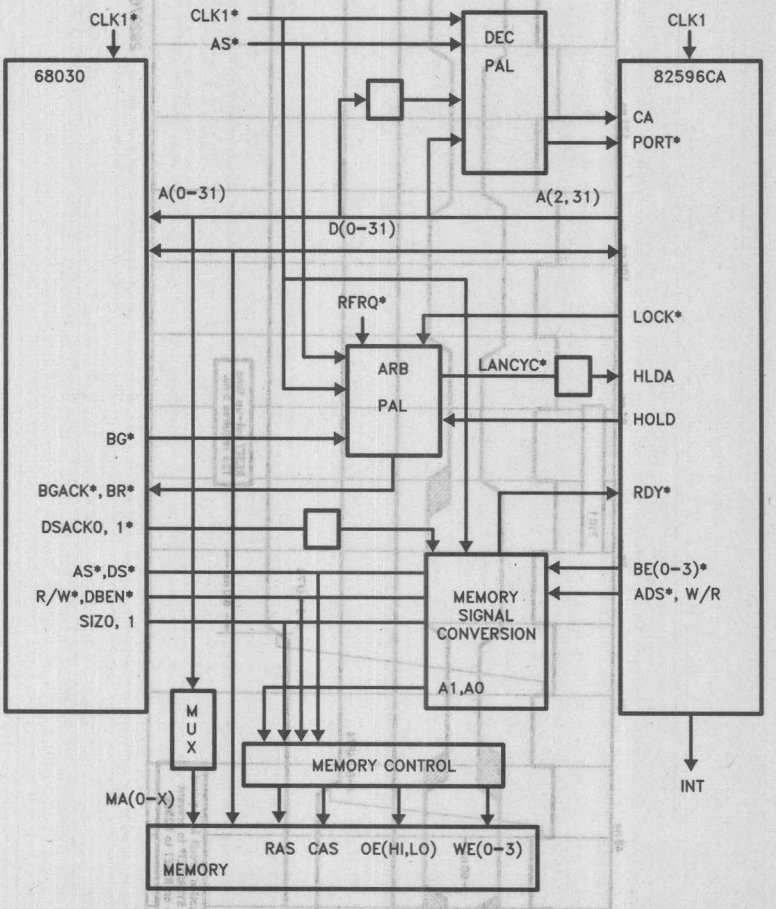
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.

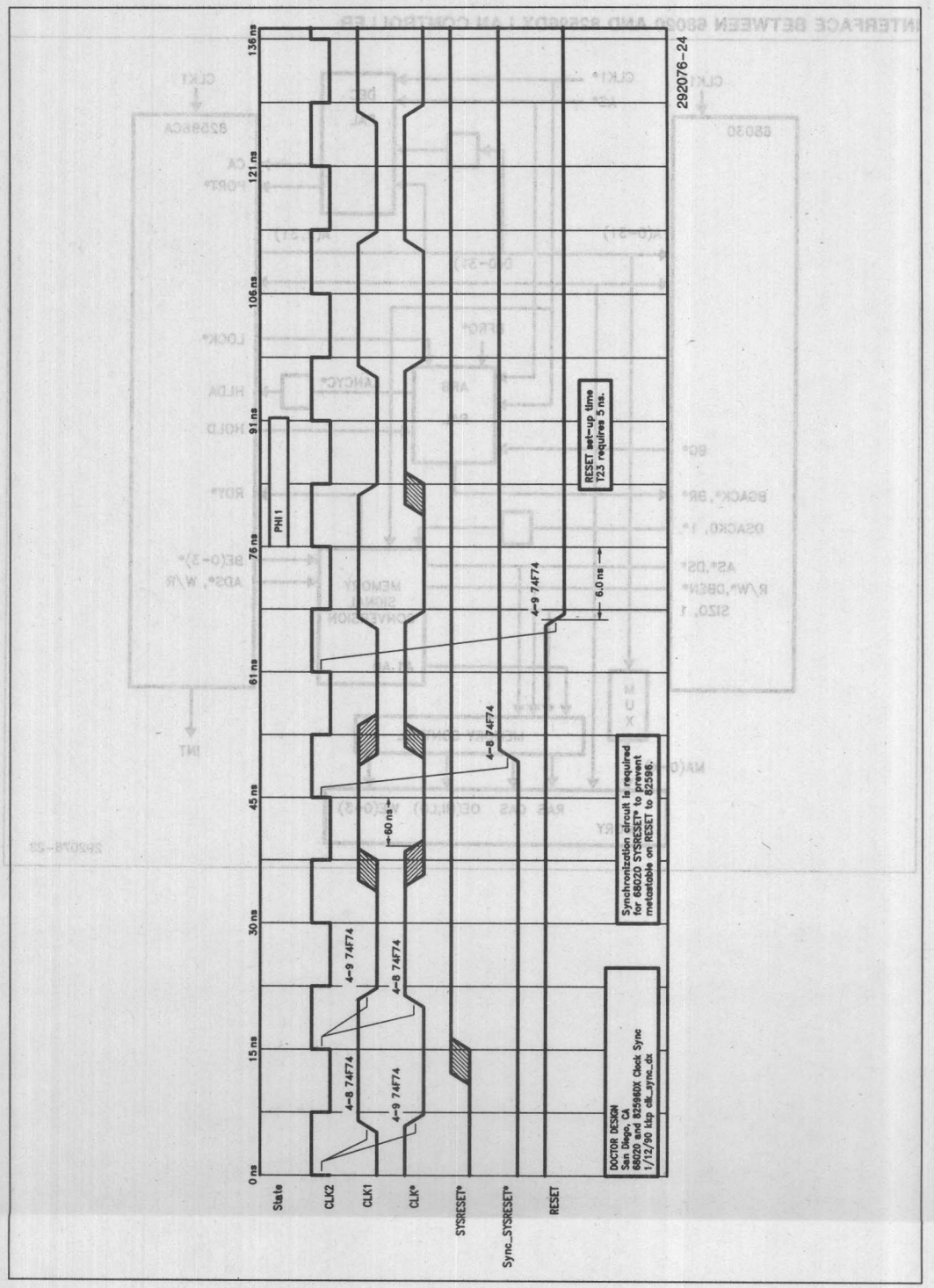
N/A = Not Applicable

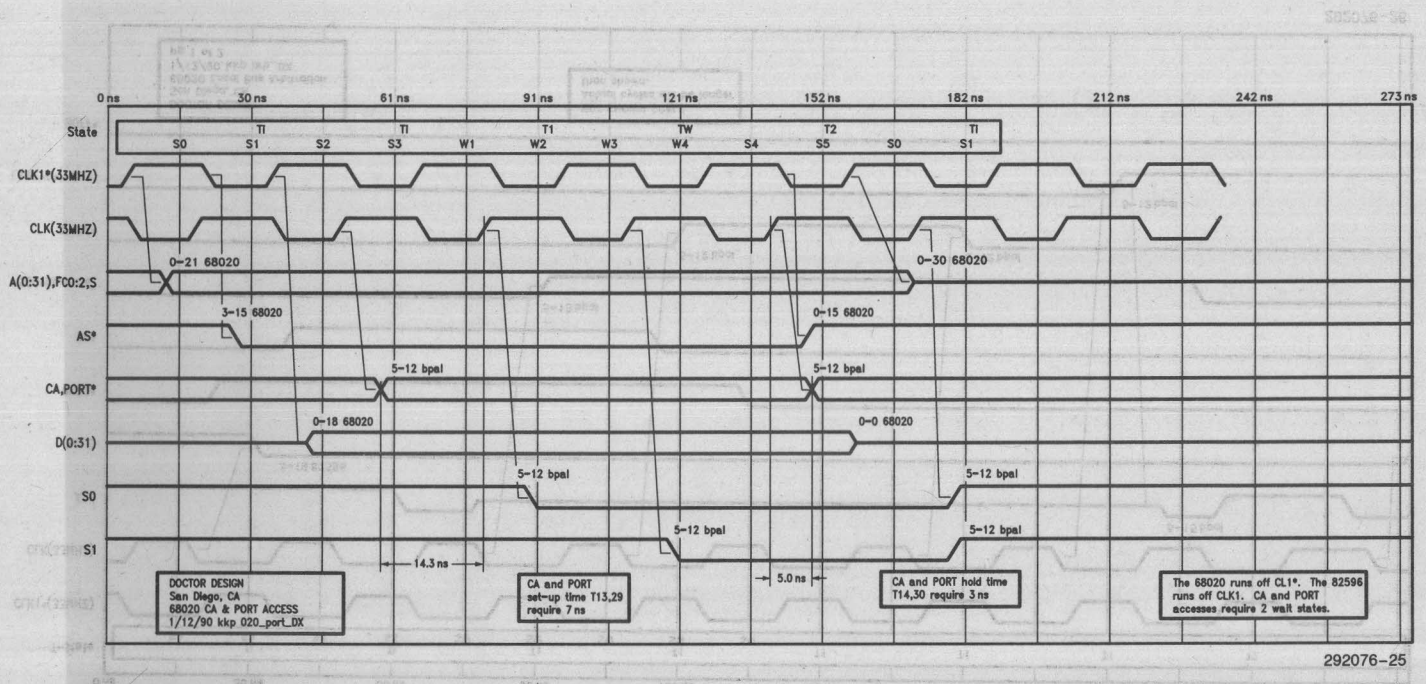
15 ns = ½ clock period

30 ns = 1 clock period

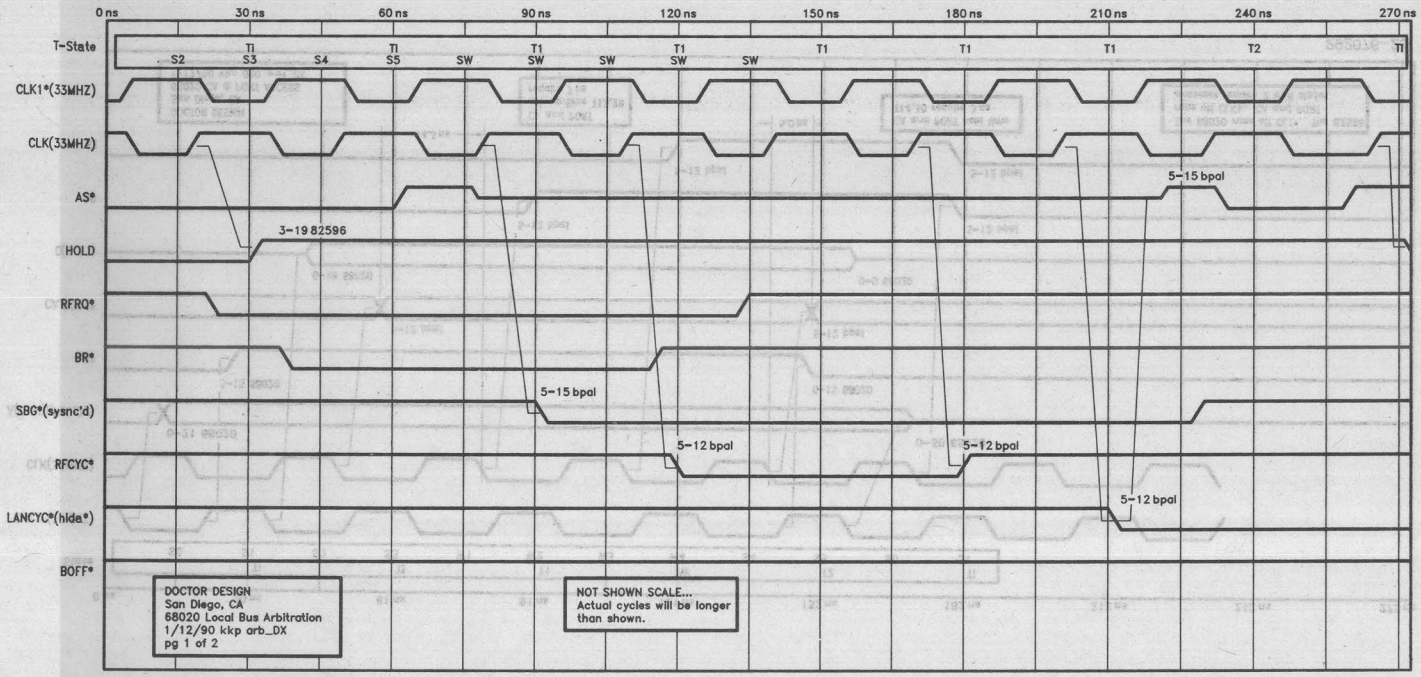
INTERFACE BETWEEN 68020 AND 82596DX LAN CONTROLLER



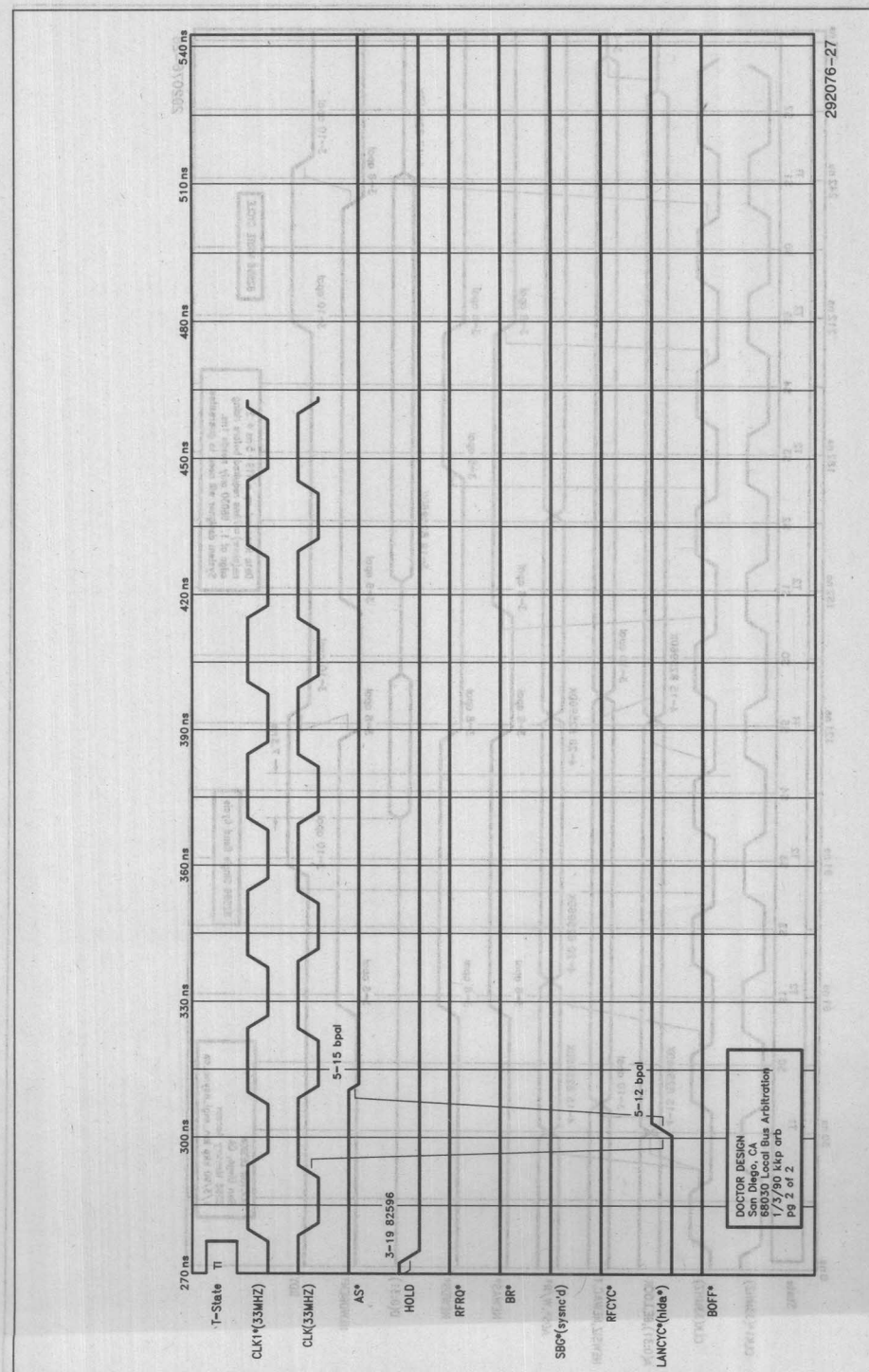

292076-24
0 ns
15 ns
30 ns
45 ns
61 ns
76 ns
91 ns
106 ns
121 ns
136 ns
State
CLK2
CLK1
CLK*
SYSRESET*
Sync_SYSRESET*
RESET
PHI 1
4-8 74F74
4-9 74F74
4-8 74F74
4-9 74F74
4-9 74F74
4-8 74F74
60 ns
6.0 ns
RESET set-up time 123 requires 5 ns.
Synchronization circuit is required for 80220 SYSRESET* to prevent metastable on RESET to 82596.
DOCTOR DESIGN
San Diego, CA
80220 and 82596DX Clock Sync
1/12/90 ikp clk_sync.dx

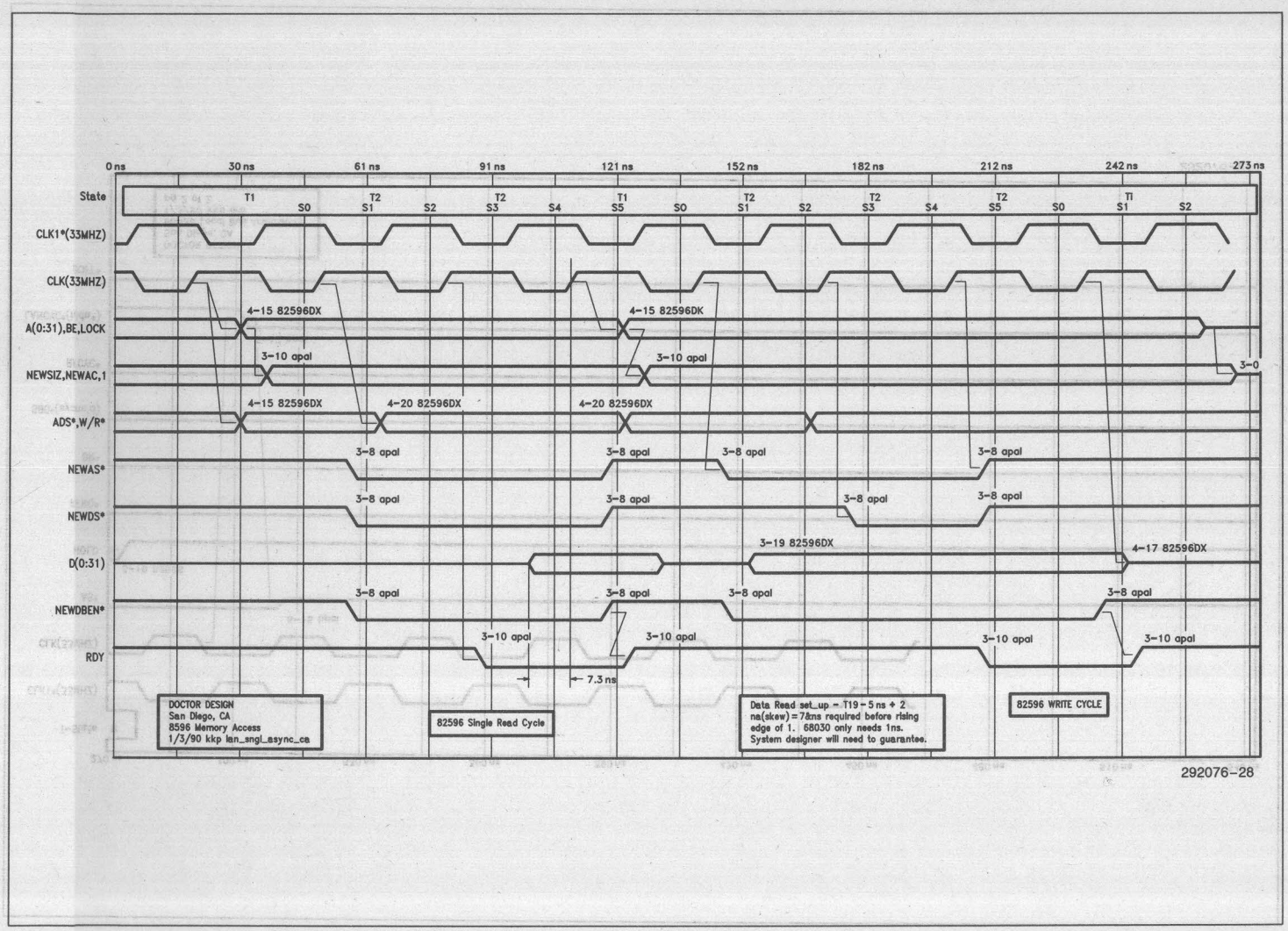

0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
TI
T1
TW
T2
TI
S0
S1
S2
S3
W1
W2
W3
W4
S4
S5
S0
S1
CLK1°(33MHZ)
CLK(33MHZ)
A(0:31),FC0:2,S
0-21 68020
0-30 68020
AS°
3-15 68020
0-15 68020
CA,PORT°
5-12 bpal
5-12 bpal
D(0:31)
0-18 68020
0-0 68020
S0
5-12 bpal
5-12 bpal
S1
5-12 bpal
5-12 bpal
14.3 ns
5.0 ns
DOCTOR DESIGN
San Diego, CA
68020 CA & PORT ACCESS
1/12/90 kkp 020_port_DX
CA and PORT
set-up time T13,29
require 7 ns
CA and PORT hold time
T14,30 require 3 ns
The 68020 runs off CL1°. The 82596
runs off CLK1. CA and PORT
accesses require 2 wait states.

292076-25


0 ns
30 ns
60 ns
90 ns
120 ns
150 ns
180 ns
210 ns
240 ns
270 ns
T-State
S2
TI
S3
S4
TI
S5
SW
T1
SW
SW
T1
SW
SW
T1
T1
T1
T2
TI
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBG*(sysnc'd)
RFCYC*
LANCYC*(hide*)
BOFF*
3-19 82596
5-15 bpal
5-15 bpal
5-12 bpal
5-12 bpal
5-12 bpal
DOCTOR DESIGN
San Diego, CA
68020 Local Bus Arbitration
1/12/90 kkp arb_DX
pg 1 of 2
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.

292076-26


270 ns
300 ns
330 ns
360 ns
390 ns
420 ns
450 ns
480 ns
510 ns
540 ns
T-State
Ti
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBC*(sysnc'd)
RFCYC*
LANCYC*(hlda*)
BOFF*
5-15 bpal
3-19 82596
5-12 bpal
DOCTOR DESIGN
San Diego, CA
68030 Local Bus Arbitration
1/3/90 kkp arb
pg 2 of 2
292076-27


0 ns
30 ns
61 ns
91 ns
121 ns
152 ns
182 ns
212 ns
242 ns
273 ns
State
CLK1*(33MHZ)
CLK(33MHZ)
A(0:31),BE,LOCK
NEWSIZ,NEWAC,1
ADS*,W/R*
NEWAS*
NEWDS*
D(0:31)
NEWDBEN*
RDY
4-15 82596DX
4-20 82596DX
3-10 apal
3-8 apal
3-19 82596DX
4-17 82596DX
7.3 ns
DOCTOR DESIGN
San Diego, CA
8596 Memory Access
1/3/90 kkp lan_sngl_async_ca
82596 Single Read Cycle
Data Read set_up = T19 - 5 ns + 2 ns(skew) = 7&ns required before rising edge of 1. 68030 only needs 1ns. System designer will need to guarantee.
82596 WRITE CYCLE
292076-28

MC68020 AND 82596DX TIMING SUMMARY FOR 33 MHz

MC68020 PARAMETERS

6 82596 puts address out 1 clock phase before 68020 S0.

6A $\overline{ADS}$ used to generate $\overline{ECS}$ and $\overline{OCS}$ before $\overline{AS}$ asserted.

7 $\overline{LANCYC}$ off + buffer off = 15 + 10 = 25 ns.

9 Derived from PLD with clock to Q delay of 8 ns.

12 Derived from PLD with clock to Q delay of 8 ns.

12A Worst case could hold $\overline{ECS}$ and $\overline{OCS}$ as long as 20 ns (82596) + delay through buffer. Note to system designer.

13 Could be a violation for $\overline{AS}$, $\overline{DS}$ to address hold of 4 ns (82596) = 8 ns (PLD) − 2 ns (skew) = −6 ns. System designer must guarantee address hold.

14 30 ns + 30 ns − 3 ns (common path through PLD) = 57 ns.

15 30 ns − 2 ns (skew) = 28 ns.

16 Floated with $\overline{LANCYC}$ going high. Minimum 30 ns to next cycle.

17 R/W invalid 1 clock cycle after $\overline{AS}$/$\overline{DS}$ negated.

18 Set with addresses from 82596.

20 Set with addresses from 82596.

21 Setting with $\overline{AS}$ could violate read cycle timing. System designer must guarantee that 5 ns setup is not required.

22 Write cycle minimum setup to $\overline{DS}$ = 30 ns − 8 ns (R/W through PLD) + 5 ns ($\overline{DS}$ through common PLD) = 27 ns. The system designer must verify that this meets memory controller timing.

23 82596 provides required time.

25 Minimum time = 30 ns (clock) − 8 ns ($\overline{AS}$ through PLD) + 4 ns (82596) = 26 ns.

25A 30 ns − 2 ns (skew) = 28 ns.

26 30 ns − 19 ns (82596) + 3 ns (PLD) = 14 ns.

27 Memory controller must guarantee 5 ns.

28 N/A

29 30 ns + 4 ns (82596) − 8 ns ($\overline{AS}$ through PLD) = 26 ns.

31 N/A

32 Plenty of time

33 Latched in ARB PLD

34 Latched in ARB PLD

35 30 ns + 15 ns = 45 ns (1.5 clocks).

37 30 ns + 15 ns = 45 ns (1.5 clocks).

37A 30 ns in ARB PLD = 1 clock

40 Asserted with $\overline{AS}$, maximum of 8 ns into clock low cycle. This should meet requirements, system designer should verify.

41 Negated in PLD, maximum 8 ns.

42 Asserted in PLD, maximum 8 ns.

43 Negated in PLD, maximum 8 ns.

44 Asserted with R/W in PLD. System designer will need to verify that 5 ns setup is not required.

45 Read = 60 ns − 2 ns (skew) = 58 ns. Write = 90 ns − 2 ns = 88 ns.

46 90 ns − 2 ns = 88 ns.

53 Data out from 82596 held valid for extra clock cycle to guarantee.

82596DX PARAMETERS

T13 30 ns − 12 ns (PLD) = 18 ns.

T14 5 ns through PLD for PORT.

T17 30 ns − 10 ns (PLD) = 20 ns.

T18 3 ns ($\overline{DBEN}$ through PLD) + 3 ns (PLD) = 6 ns.

T19 May violate by 5 ns + 2 ns (skew) − 5 ns (memory controller) = 2 ns. System designer will need to guarantee extra 2 ns setup time.

T20 3 ns ($\overline{DS}$ from PLD) + delay through memory controller.

T21 30 ns − 8 ns = 22 ns.

T22 3 ns (PLD) + external inverter.

T26 3 CLK2 cycles.

T27 30 ns + 15 ns − 18 ns (68020) = 27 ns

T29 30 ns − 12 ns = 18 ns.

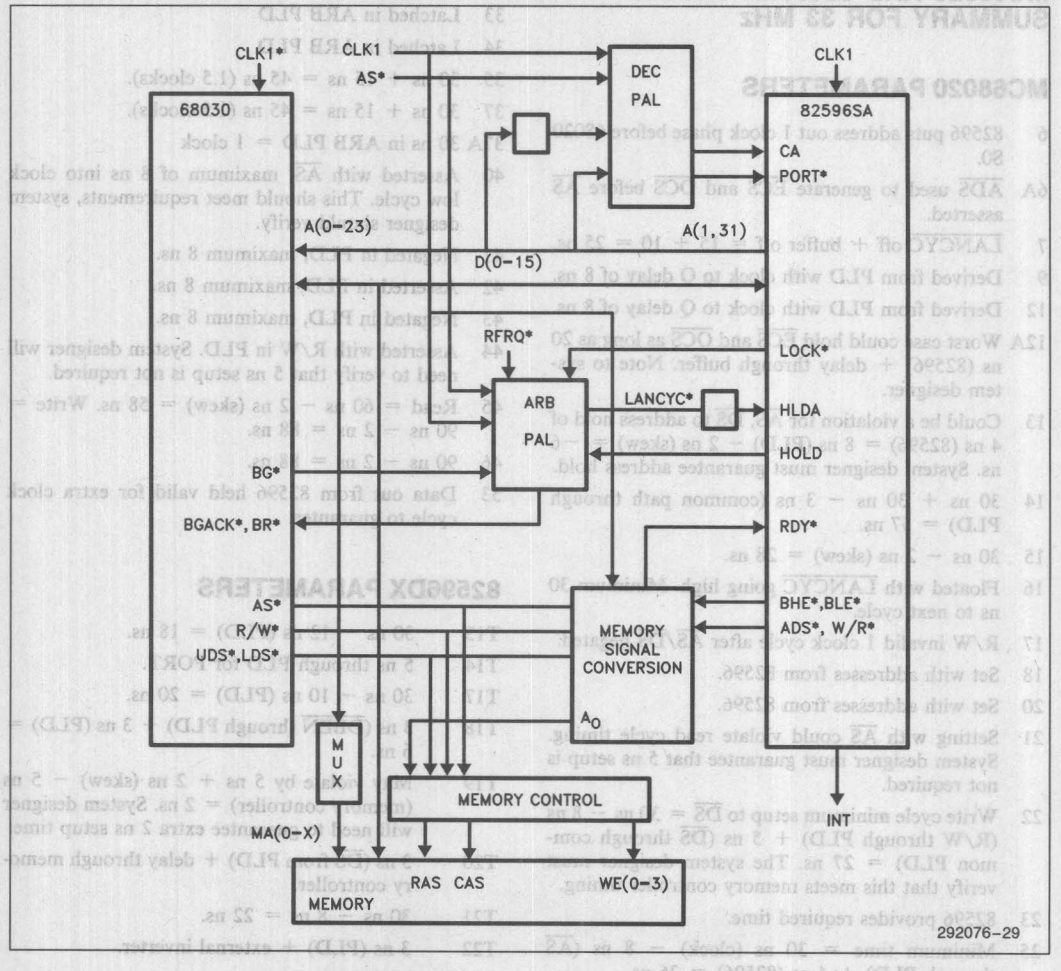
T30 Minimum 3 ns through 10, 12, or 15 ns PLD.

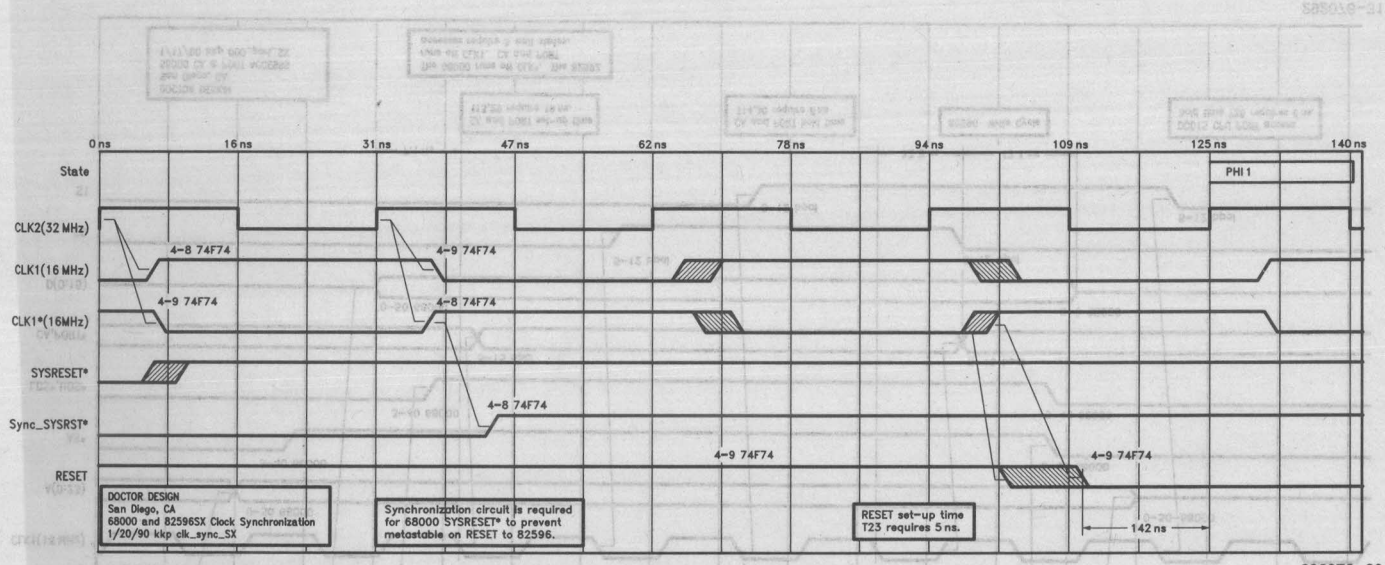
N/A = Not Applicable

15 ns = ½ clock period

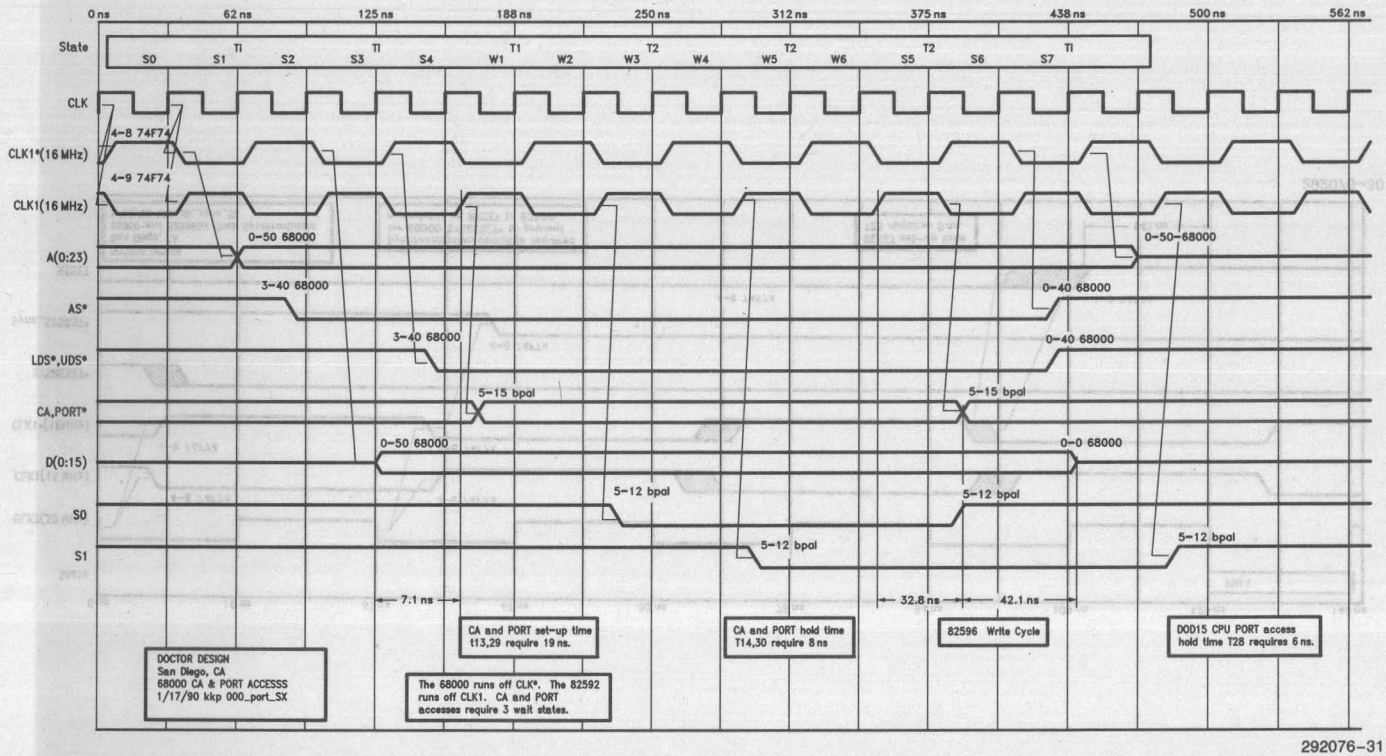
30 ns = 1 clock period

INTERFACE BETWEEN 68000 AND 82596SX LAN CONTROLLER

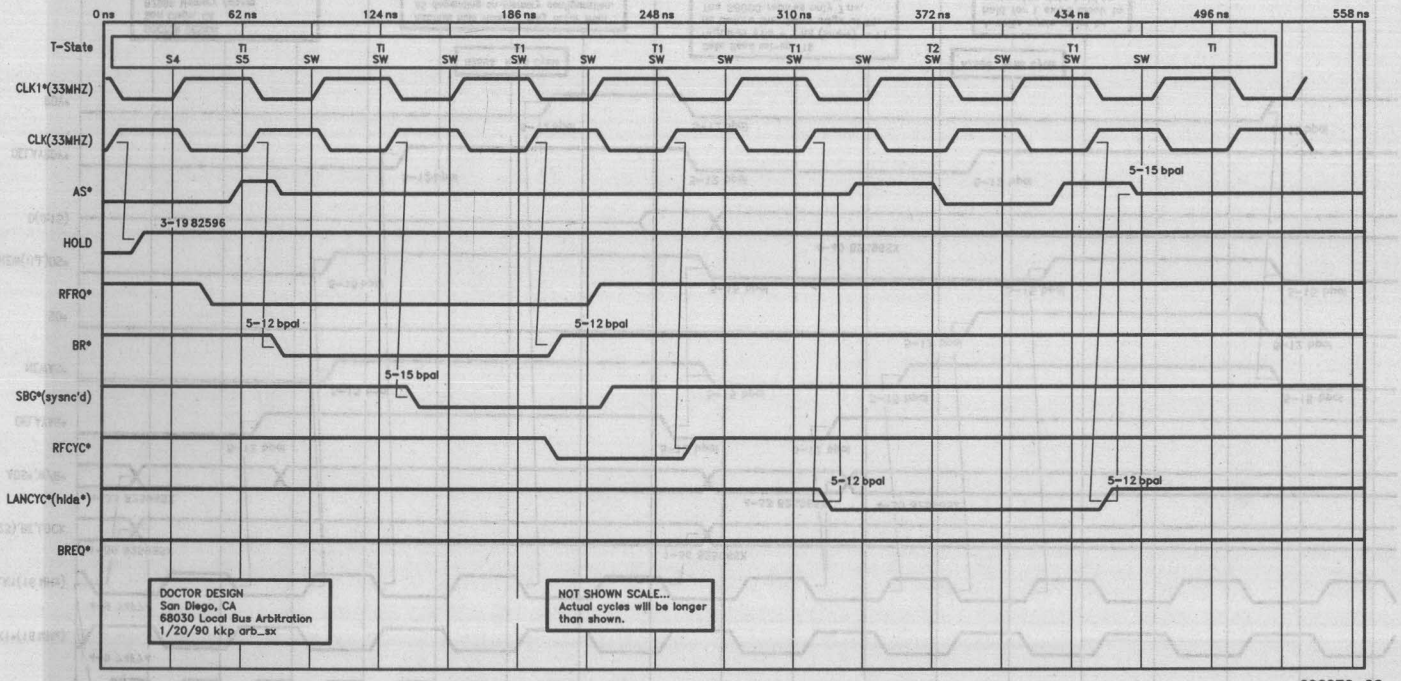



0 ns
16 ns
31 ns
47 ns
62 ns
78 ns
94 ns
109 ns
125 ns
140 ns
State
PHI 1
CLK2(32 MHz)
CLK1(16 MHz)
CLK1*(16MHz)
SYSRESET*
Sync_SYSRST*
RESET
4-8 74F74
4-9 74F74
4-9 74F74
4-8 74F74
4-8 74F74
4-9 74F74
4-9 74F74
142 ns
DOCTOR DESIGN
San Diego, CA
68000 and 82596SX Clock Synchronization
1/20/90 kkp clk_sync_SX
Synchronization circuit is required for 68000 SYSRESET* to prevent metastable on RESET to 82596.
RESET set-up time T23 requires 5 ns.

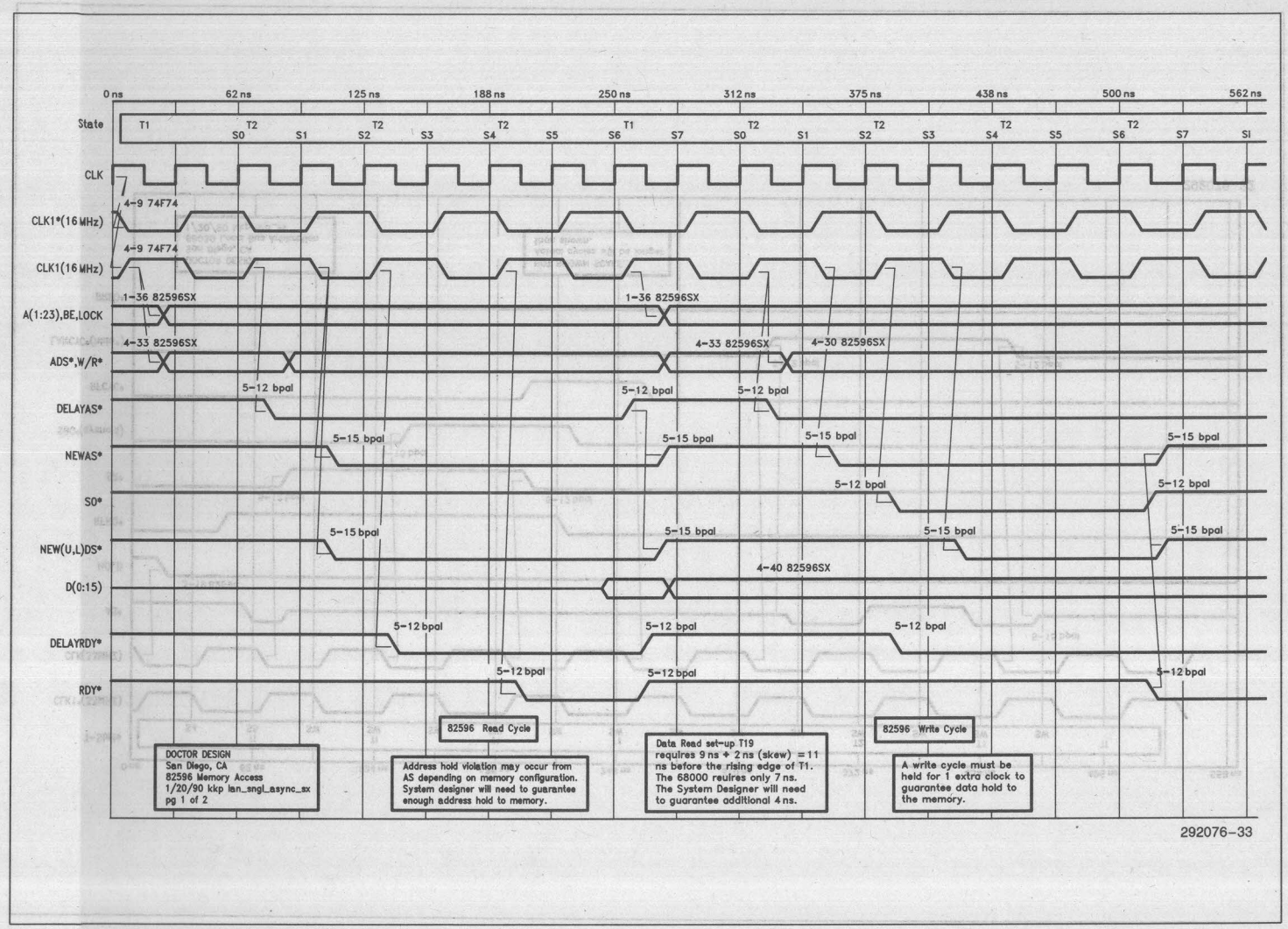
292076-30


0 ns
62 ns
125 ns
188 ns
250 ns
312 ns
375 ns
438 ns
500 ns
562 ns
State
S0 S1 TI S2 S3 TI S4 W1 T1 W2 W3 T2 W4 W5 T2 W6 S5 T2 S6 S7 TI
CLK
CLK1*(16 MHz)
4-8 74F74
CLK1(16 MHz)
4-9 74F74
A(0:23)
0-50 68000
0-50-68000
AS*
3-40 68000
0-40 68000
LDS*,UDS*
3-40 68000
0-40 68000
CA,PORT*
5-15 bpal
5-15 bpal
D(0:15)
0-50 68000
0-0 68000
S0
5-12 bpal
5-12 bpal
S1
5-12 bpal
5-12 bpal
7.1 ns
32.8 ns
42.1 ns
DOCTOR DESIGN
San Diego, CA
68000 CA & PORT ACCESSS
1/17/90 kkp 000_port_SX
CA and PORT set-up time
t13,29 require 19 ns.
The 68000 runs off CLK*. The 82592
runs off CLK1. CA and PORT
accesses require 3 wait states.
CA and PORT hold time
T14,30 require 8 ns
82596 Write Cycle
DOD15 CPU PORT access
hold time T28 requires 6 ns.

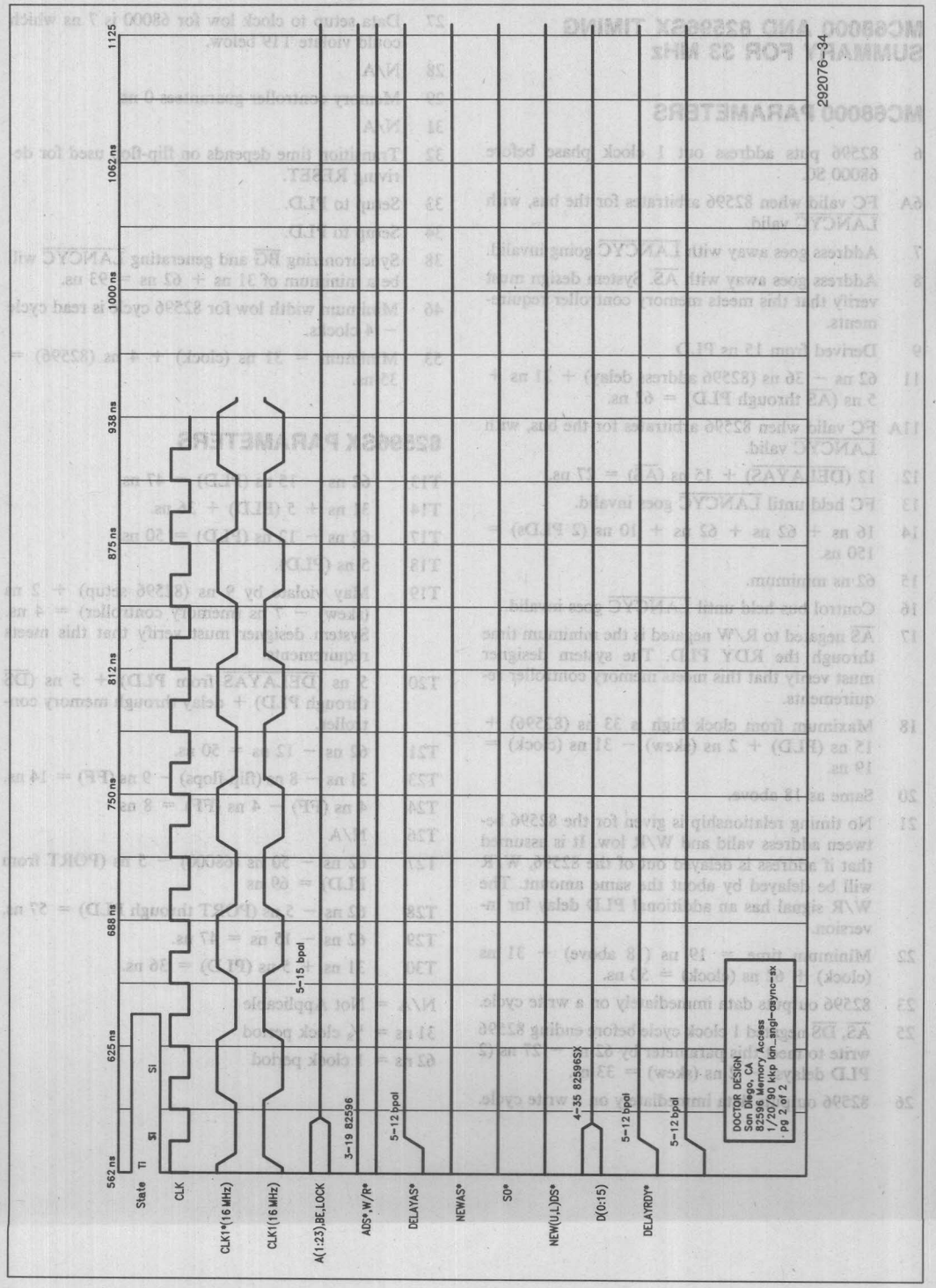
292076-31


0 ns
62 ns
124 ns
186 ns
248 ns
310 ns
372 ns
434 ns
496 ns
558 ns
T-State
S4
TI
S5
SW
TI
SW
T1
SW
T2
SW
TI
CLK1*(33MHZ)
CLK(33MHZ)
AS*
HOLD
RFRQ*
BR*
SBG*(sysnc'd)
RFCYC*
LANCYC*(hlda*)
BREQ*
3-19 82596
5-12 bpal
5-15 bpal
DOCTOR DESIGN
San Diego, CA
68030 Local Bus Arbitration
1/20/90 kkp arb_sx
NOT SHOWN SCALE...
Actual cycles will be longer
than shown.

292076-32


0 ns
62 ns
125 ns
188 ns
250 ns
312 ns
375 ns
438 ns
500 ns
562 ns
State
T1
T2
S0
S1
S2
S3
S4
S5
S6
S7
T1
T2
S0
S1
S2
S3
S4
S5
S6
S7
SI
CLK
CLK1*(16 MHz)
4-9 74F74
CLK1(16 MHz)
4-9 74F74
A(1:23),BE,LOCK
1-36 82596SX
1-36 82596SX
ADS*,W/R*
4-33 82596SX
4-33 82596SX
4-30 82596SX
DELAYAS*
5-12 bpal
5-12 bpal
5-12 bpal
NEWAS*
5-15 bpal
5-15 bpal
5-15 bpal
5-15 bpal
S0*
5-12 bpal
5-12 bpal
NEW(U,L)DS*
5-15 bpal
5-15 bpal
5-15 bpal
5-15 bpal
D(0:15)
4-40 82596SX
DELAYRDY*
5-12 bpal
5-12 bpal
5-12 bpal
RDY*
5-12 bpal
5-12 bpal
5-12 bpal
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/20/90 kkp lan_sngl_async_sx
pg 1 of 2
82596 Read Cycle
Address hold violation may occur from AS depending on memory configuration. System designer will need to guarantee enough address hold to memory.
Data Read set-up T19 requires 9 ns + 2 ns (skew) = 11 ns before the rising edge of T1. The 68000 realires only 7 ns. The System Designer will need to guarantee additional 4 ns.
82596 Write Cycle
A write cycle must be held for 1 extra clock to guarantee data hold to the memory.

292076-33


562 ns
625 ns
688 ns
750 ns
812 ns
875 ns
938 ns
1000 ns
1062 ns
1125
State
T1
SI
SI
CLK
CLK1*(16 MHz)
CLK1(16 MHz)
A(1:23),BE,LOCK
ADS*,W/R*
DELAYAS*
NEWAS*
S0*
NEW(U,L)DS*
D(0:15)
DELAYRDY*
5–15 bpol
3–19 82596
5–12 bpol
4–35 82596SX
5–12 bpol
5–12 bpol
DOCTOR DESIGN
San Diego, CA
82596 Memory Access
1/20/90 kkp ian_angl–async–sx
pg 2 of 2
292076–34

MC68000 AND 82596SX TIMING SUMMARY FOR 33 MHz

MC68000 PARAMETERS

6 82596 puts address out 1 clock phase before 68000 S0.

6A FC valid when 82596 arbitrates for the bus, with $\overline{\text{LANCYC}}$ valid.

7 Address goes away with $\overline{\text{LANCYC}}$ going invalid.

8 Address goes away with $\overline{\text{AS}}$. System design must verify that this meets memory controller requirements.

9 Derived from 15 ns PLD.

11 62 ns − 36 ns (82596 address delay) + 31 ns + 5 ns ($\overline{\text{AS}}$ through PLD) = 62 ns.

11A FC valid when 82596 arbitrates for the bus, with $\overline{\text{LANCYC}}$ valid.

12 12 ($\overline{\text{DELAYAS}}$) + 15 ns ($\overline{\text{AS}}$) = 27 ns.

13 FC held until $\overline{\text{LANCYC}}$ goes invalid.

14 16 ns + 62 ns + 62 ns + 10 ns (2 PLDs) = 150 ns.

15 62 ns minimum.

16 Control bus held until $\overline{\text{LANCYC}}$ goes invalid.

17 $\overline{\text{AS}}$ negated to R/W negated is the minimum time through the RDY PLD. The system designer must verify that this meets memory controller requirements.

18 Maximum from clock high is 33 ns (82596) + 15 ns (PLD) + 2 ns (skew) − 31 ns (clock) = 19 ns.

20 Same as 18 above.

21 No timing relationship is given for the 82596 between address valid and W/R low. It is assumed that if address is delayed out of the 82596, W/R will be delayed by about the same amount. The W/R signal has an additional PLD delay for inversion.

22 Minimum time = 19 ns (18 above) − 31 ns (clock) + 62 ns (clock) = 50 ns.

23 82596 outputs data immediately on a write cycle.

25 $\overline{\text{AS}}$, $\overline{\text{DS}}$ negated 1 clock cycle before ending 82596 write to meet this parameter by 62 ns − 27 ns (2 PLD delays) − 2 ns (skew) = 33 ns.

26 82596 outputs data immediately on a write cycle.

27 Data setup to clock low for 68000 is 7 ns which could violate T19 below.

28 N/A

29 Memory controller guarantees 0 ns.

31 N/A

32 Transition time depends on flip-flop used for deriving RESET.

33 Setup to PLD.

34 Setup to PLD.

38 Synchronizing $\overline{\text{BG}}$ and generating $\overline{\text{LANCYC}}$ will be a minimum of 31 ns + 62 ns = 93 ns.

46 Minimum width low for 82596 cycle is read cycle − 4 clocks.

53 Minimum = 31 ns (clock) + 4 ns (82596) = 35 ns.

82596SX PARAMETERS

T13 62 ns − 15 ns (PLD) = 47 ns.

T14 31 ns + 5 (PLD) + 36 ns.

T17 62 ns − 12 ns (PLD) = 50 ns.

T18 5 ns (PLD).

T19 May violate by 9 ns (82596 setup) + 2 ns (skew) − 7 ns (memory controller) = 4 ns. System designer must verify that this meets requirements.

T20 5 ns ($\overline{\text{DELAYAS}}$ from PLD) + 5 ns ($\overline{\text{DS}}$ through PLD) + delay through memory controller.

T21 62 ns − 12 ns = 50 ns.

T23 31 ns − 8 ns (flip-flops) − 9 ns (FF) = 14 ns.

T24 4 ns (FF) − 4 ns (FF) = 8 ns

T26 N/A

T27 62 ns − 50 ns (68000) − 5 ns (PORT from PLD) = 69 ns

T28 62 ns − 5 ns (PORT through PLD) = 57 ns.

T29 62 ns − 15 ns = 47 ns.

T30 31 ns + 5 ns (PLD) = 36 ns.

N/A = Not Applicable

31 ns = ½ clock period

62 ns = 1 clock period

APPENDIX D
PARTS LISTS

Each parts list includes only those components that are part of the interface. The memory controller and memory components are not included.

D.1 MC68030/82596CA

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
3	20R4	24-pin PLD; 4 Registered Outputs
1	20R8	24-pin PLD; 8 Registered Outputs
0.5	74F244	Octal Tri-State Buffer

Each PLD must have no more than 10 ns propagation delay for 33 MHz design.
Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

D.2 MC68020/82596DX

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
2	20R4	24-pin PLD; 4 Registered Outputs
1	20R6	24-pin PLD; 6 Registered Outputs
0.5	74F244	Octal Tri-State Buffer

Each PLD must have no more than 10 ns propagation delay for 33 MHz design.
Each PLD must have no more than 15 ns propagation delay for 25 MHz design.

D.3 MC68000/82596SX

Quantity	Generic Number	Description
1.5	74F74	Dual D Flip-Flop
2	20R4	24-pin PLD; 4 Registered Outputs
1	20R8	24-pin PLD; 8 Registered Outputs

Each PLD must have no more than 15 ns propagation delay for 16 MHz design.

October 1990

Implementing 10Base-T Networks with Intel's Twisted Pair Ethernet* Components and Supercomponents

WILLIAM WAGER
TECHNICAL MARKETING ENGINEER

*Ethernet is a registered trademark of Xerox Corporation.

Order Number: 292080-001

Implementing 10Base-T Networks with Intel's Twisted Pair Ethernet Components and Supercomponents

CONTENTS

PAGE

PROLOGUE 1-572

1.0 INTRODUCTION 1-572

2.0 SYSTEM DESCRIPTION 1-574

2.1 Network Description 1-574

2.1.1 Medium Attachment Unit 1-574

2.1.2 Multiple Port Repeaters 1-575

2.1.3 Data Terminal Equipment ... 1-576

2.1.4 Link Segments 1-576

2.2 Integrating with Existing 802.3 Networks 1-576

2.3 Software Compatibility 1-577

3.0 NETWORK SYSTEM COMPONENT DESIGN 1-577

3.1 Designing a DTE Node Based on the 82521TB Serial Supercomponent 1-577

3.2 Designing a DTE Node Based on the 82506TB Twisted Pair MAU Chip 1-577

3.2.1 Host to Ethernet LAN Controller Interface 1-579

3.2.2 Ethernet LAN Controller to 82C501AD Interface 1-579

3.2.3 The 82C501AD to 82506TB Interface 1-579

3.2.4 The 82506TB to Analog Front-End Interface (AFE) 1-579

CONTENTS

PAGE

3.3 Using the 82523TB Twisted Pair Ethernet MAU to Convert Existing Ethernet Nodes to 10BASE-T 1-579

3.4 Designing a MAU Using the 82506TB Twisted Pair MAU Chip .. 1-579

3.5 Designing a Multiple Port Repeater Based on the 82505TA MPR 1-579

3.5.1 82505TA to 82504TA Interface and Clock Generation 1-579

3.5.2 Twisted Pair Port Design 1-581

3.5.3 AUI Port 1-581

3.5.4 Port Disable Control 1-581

3.5.5 LED Control 1-581

4.0 ANALOG FRONT END 1-584

4.1 Preconditioning Voltage Summer 1-584

4.2 High-Voltage Protection 1-584

4.3 EMI Filter 1-584

4.4 Line Coupling Devices 1-584

4.5 Layout Considerations 1-586

5.0 SUMMARY 1-587

APPENDIX A. EQUATIONS FOR EPLD VERSION OF 10BASE-T STATE MACHINES 1-588

PROLOGUE

At the time of this publication, the IEEE 802.3's 10BASE-T task force is completing its work writing the Twisted Pair Ethernet standard, and the IEEE 802.3 working group has approved the standard. The draft is finished, all that remains is for the IEEE Technical Committee on Computer Communications to approve the draft and to forward it to the IEEE Standards Board, which is expected in September 1990. The work now falls to the designers of 10BASE-T products to implement the standard correctly. The objective of this application note is to aid system designers to meet this goal.

1.0 INTRODUCTION

This application note is presented as an aid to designing a twisted pair Ethernet (TPE) LAN using Intel's TPE products and Intel's family of Ethernet LAN controllers (82586, 82590, and the 82596 family). It is aimed at system designers working on TPE designs who have working knowledge of the IEEE 802.3 standards and some analog design expertise. It can be used in conjunction with application note AP-274, *Implementing Ethernet/ Cheapernet with the Intel 82586*, by Kiyoshi Nishide. It supersedes AP-324 *Implementing Twisted Pair Ethernet with the Intel 82504TA, 82505TA, and 82521TA*.

Intel has introduced the 82521TB Serial Supercomponent (SSC), the 82523TB Medium Attachment Unit (MAU), the 82505TA Multiport Repeater Controller (MPR), and the 82506TB Twisted Pair MAU Chip (TPMC). They simplify the design of Twisted Pair Ethernet LANs based on the 10BASE-T standard. The 10BASE-T network is compatible with other ANSI/ IEEE 802.3 networks (e.g., 10BASE5 and 10BASE2) at the medium access control (MAC) and physical signaling sublayers. This means that a twisted pair Ethernet LAN based on these products will be software compatible with 802.3 standard networks, and can be included in mixed networks by connection through a standard attachment unit interface (AUI) port of a repeater.

A twisted pair Ethernet LAN using these products consists of several elements: data terminal equipment (DTE), medium attachment units (MAU), multiple port repeaters (MPR), and a cable plant. More complex networks, which connect to existing 802.3 networks (e.g., 10BASE5 or 10BASE2), can be constructed by using the 802.3-standard AUI port of the MPR. A typical network using all of these elements is shown in Figure 1.

Four types of DTE and MAU combinations are shown. Two are embedded MAUs (contained within the DTE) and two are external MAUs (connected to the DTE node through a standard AUI cable). The embedded MAU designs use either the 82521TB SSC or the 82506TB TP MAU chip and its associated circuitry. One external MAU is the 82523TB, the other is based on the 82506TB. The repeaters are designed around the 82505TA and also contain one 82504TA. Each repeater contains 11 twisted pair ports, with embedded MAUs, and 1 AUI port.

The cable plant consists of standard telephone wire: either 4- or 25-pair, unshielded, twisted pair (26–22 gauge). Each segment uses two twisted pairs, transmit (TD) and receive (RD), and can coexist with other services (such as standard telephone) in the same cable bundle. Each segment has a maximum length of 100 m.

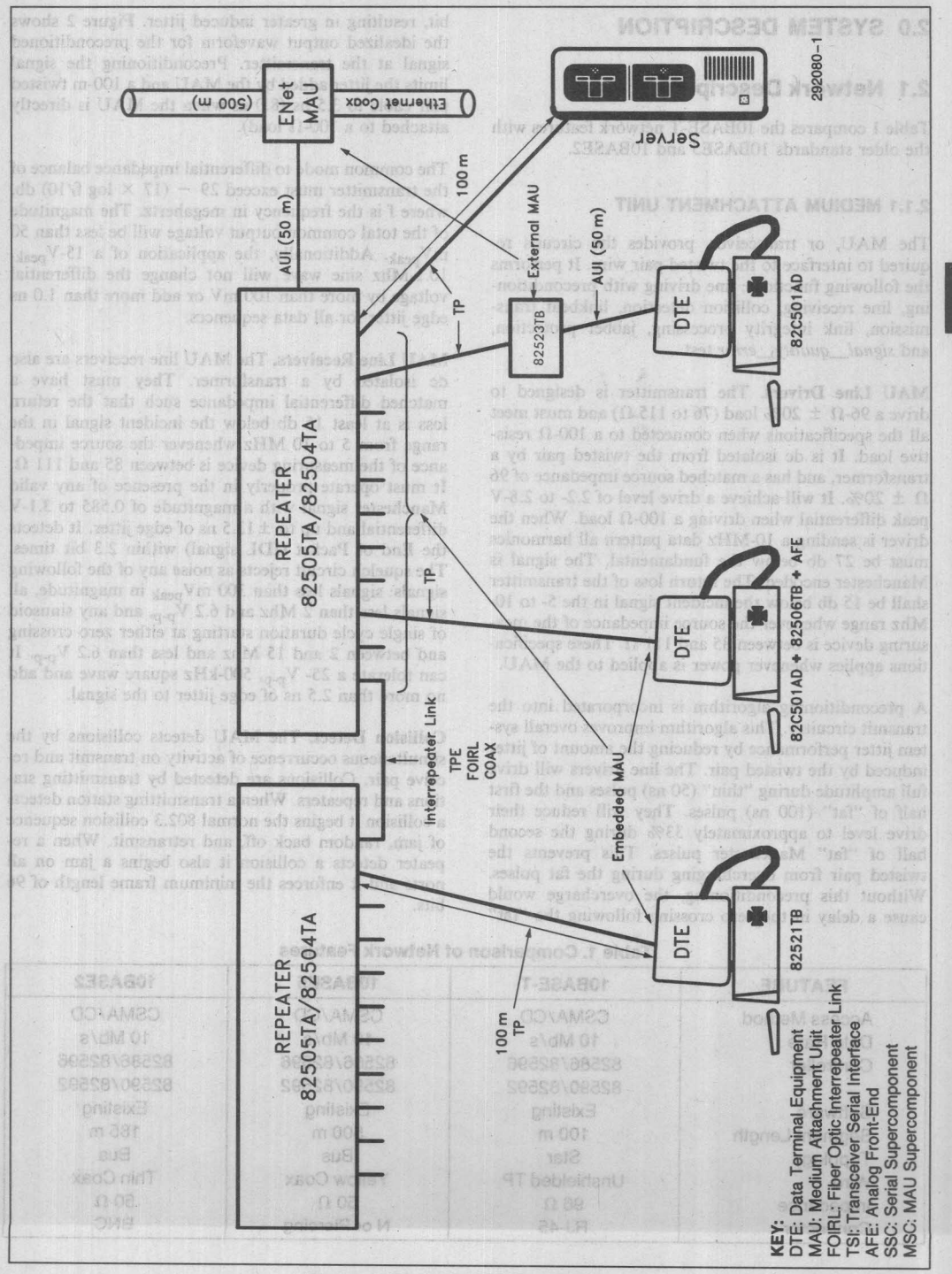


Figure 1. Typical 10BASE-T Network

2.0 SYSTEM DESCRIPTION

2.1 Network Description

Table 1 compares the 10BASE-T network features with the older standards 10BASE5 and 10BASE2.

2.1.1 MEDIUM ATTACHMENT UNIT

The MAU, or transceiver, provides the circuits required to interface to the twisted pair wire. It performs the following functions: line driving with preconditioning, line receiving, collision detection, linkbeat transmission, link integrity processing, jabber protection, and *signal_quality_error* test.

MAU Line Drivers. The transmitter is designed to drive a 96-Ω ± 20% load (76 to 115 Ω) and must meet all the specifications when connected to a 100-Ω resistive load. It is dc isolated from the twisted pair by a transformer, and has a matched source impedance of 96 Ω ± 20%. It will achieve a drive level of 2.2- to 2.8-V peak differential when driving a 100-Ω load. When the driver is sending a 10-MHz data pattern all harmonics must be 27 db below the fundamental. The signal is Manchester encoded. The return loss of the transmitter shall be 15 db below the incident signal in the 5- to 10-Mhz range whenever the source impedance of the measuring device is between 85 and 111 Ω. These specifications applies whenever power is applied to the MAU.

A preconditioning algorithm is incorporated into the transmit circuitry. This algorithm improves overall system jitter performance by reducing the amount of jitter induced by the twisted pair. The line drivers will drive full amplitude during "thin" (50 ns) pulses and the first half of "fat" (100 ns) pulses. They will reduce their drive level to approximately 33% during the second half of "fat" Manchester pulses. This prevents the twisted pair from overcharging during the fat pulses. Without this preconditioning, the overcharge would cause a delay in the zero crossing following the "fat" bit, resulting in greater induced jitter. Figure 2 shows the idealized output waveform for the preconditioned signal at the transmitter. Preconditioning the signal limits the jitter added by the MAU and a 100-m twisted pair cable to 3.5 ns (8.0 ns when the MAU is directly attached to a 100-Ω load).

The common mode to differential impedance balance of the transmitter must exceed $29 - (17 \times \log f/10)$ db, where f is the frequency in megahertz. The magnitude of the total common output voltage will be less than 50 mV_{peak}. Additionally, the application of a 15-V_{peak}, 10.1-Mhz sine wave will not change the differential voltage by more than 100 mV or add more than 1.0 ns edge jitter for all data sequences.

MAU Line Receivers. The MAU line receivers are also dc isolated by a transformer. They must have a matched differential impedance such that the return loss is at least 15 db below the incident signal in the range from 5 to 10 MHz whenever the source impedance of the measuring device is between 85 and 111 Ω. It must operate properly in the presence of any valid Manchester signal with a magnitude of 0.585 to 3.1-V differential and up to ± 13.5 ns of edge jitter. It detects the End of Packet (IDL signal) within 2.3 bit times. The squelch circuit rejects as noise any of the following signals: signals less than 300 mV_{peak} in magnitude, all signals less than 2 Mhz and 6.2 V_{p-p}, and any sinusoid of single cycle duration starting at either zero crossing and between 2 and 15 Mhz and less than 6.2 V_{p-p}. It can tolerate a 25- V_{p-p}, 500-kHz square wave and add no more than 2.5 ns of edge jitter to the signal.

Collision Detect. The MAU detects collisions by the simultaneous occurrence of activity on transmit and receive pair. Collisions are detected by transmitting stations and repeaters. When a transmitting station detects a collision it begins the normal 802.3 collision sequence of jam, random back off, and retransmit. When a repeater detects a collision it also begins a jam on all ports and it enforces the minimum frame length of 96 bits.

Table 1. Comparison of Network Features

FEATURE	10BASE-T	10BASE5	10BASE2
Access Method	CSMA/CD	CSMA/CD	CSMA/CD
Data Rate	10 Mb/s	10 Mb/s	10 Mb/s
Controller	82586/82596 82590/82592	82586/82596 82590/82592	82586/82596 82590/82592
Software	Existing	Existing	Existing
Segment Length	100 m	500 m	185 m
Topology	Star	Bus	Bus
Wire	Unshielded TP	Yellow Coax	Thin Coax
Impedance	96 Ω	50 Ω	50 Ω
Connector	RJ-45	N or Piercing	BNC

Figure 2. Preconditioned Waveform

Loopback. During transmission without collision, the loopback function of the 10BASE-T MAU will also route the transmitted data back to the DTE on the receive circuit. This function mimics the natural data loopback which occurs in coax MAUs.

Link Integrity. The link integrity function is a process by which the 10BASE-T MAU can determine if its receiver is properly connected to a compatible MAU. If it is not, it disables its transmitter, receiver, and loopback functions. This prevents a one-way link failure from indefinitely disrupting the network, since the carrier sense function is dependent on the receiver. When a bad link disables a MAU's carrier sense function, it removes itself from the network.

The link integrity function is accomplished by two independent and asynchronous activities—one for the transmitter and one for the receiver. The transmitter will fill long periods of idle with link test pulses (link beats). A link beat is transmitted after every 8 to 24 ms of silence. This defines a maximum period of silence the remote receiver will experience regardless of network traffic.

The receiver monitors its circuit for data packet and link beat reception from the remote MAU transmitter. If an excessively long period of silence occurs, the MAU will disable its receiver, transmitter, and loopback functions. Link beats are still transmitted in this mode. Once data or link beat reception resumes, the MAU reenables all its functions.

2.1.2 MULTIPLE PORT REPEATER

The repeater is the central point in the star configured network. It is usually located in a telephone closet or other central wiring point. The link segments (repeater to node or repeater to repeater connections) are then made by using available twisted pairs in the existing telephone cable plant or a dedicated cable plant. The repeater must conform to the ANSI/IEEE 802.3c-1988 standard for repeaters. It can have any number of dedicated 10BASE-T, 10BASE2, FOIRL, or 10BASE5 ports, and it can have any number of attachment unit interface (AUI) ports. The AUI ports are DTE type (DB-15 female receptacle) and can be connected to any valid 802.3 MAU. All dedicated 10BASE-T ports must support the same functions as the 10BASE-T MAU and normal repeater port functions.

The repeater supports autopartitioning and jabber protection. These two features prevent faulty nodes from taking down the entire network. The autopartition algorithm monitors ports for consecutive collisions such as would happen if a coax segment (10BASE5 or 10BASE2) were left unterminated or if the Tx and Rx twisted pairs were shorted together on a 10BASE-T segment. Once identified, that port is removed from the network until the fault condition is removed; this allows the remainder of the network to operate normally.

The jabber function of the repeater monitors the length of incoming data. If it detects an abnormally long frame it breaks it into legal lengths by inserting minimum interframe spaces on its transmitted signal. This prevents any jabber condition from being repeated onto other segments. Repetition of the jabber condition would allow its own, and other MAUs, to enter a fail state due to faults at a remote location, thus preventing normal operation of the network after the fault condition is removed. With the repeater's jabber protection, network operation resumes after the fault is removed.

2.1.3 DATA TERMINAL EQUIPMENT

DTEs include the user nodes, file servers, bridges, and other entities that can originate and accept data packets on the network. DTEs contain the medium access control (MAC) and physical layer signaling (PLS) sublayers. A DTE can also contain an embedded MAU. DTEs that do not have an embedded MAU have an AUI connector. DTEs with embedded MAUs have the medium dependent interface connector for that particular MAU (RJ-45 for 10BASE-T and BNC for 10BASE2). The MAC functions are handled by the LAN controller (Intel's 82586, 82590, 82592, or the 82596 family). The PLS functions are handled by the serial interface component (82C501AD) or a combination PLS/MAU device (82504TA and 82521TB). This architecture represents a continuity of design for migration from Ethernet/Cheapernet designs to Twisted Pair Ethernet. Only the MAU part of the design needs to be updated. This is 100% software independent.

2.1.4 LINK SEGMENTS

A 10BASE-T link segment connects two twisted pair MAUs; it comprises two medium dependent interface connectors (RJ-45 and 8-pin, standard telephone plugs), two pairs of twisted pair wire (not to exceed 100 m), and a crossover. The connector's pin assignments are shown in Table 2.

TABLE 2. MDI CONNECTOR PIN ASSIGNMENTS

Pin	Signal
1	Transmit Data + (TD+)
2	Transmit Data − (TD−)
3	Receive Data + (RD+)
4	Not Used
5	Not Used
6	Receive Data − (RD−)
7	Not Used
8	Not Used

The crossover function connects the TD outputs of one MAU to the RD inputs of the other. This function can be external or embedded within a MAU. If the function is embedded the signal names on the connector refer to the remote MAU. That is, pin 1 (TD+) on a MAU with an embedded crossover is connected to the Transmit Data (+) of the remote MAU and to its own Receive Data (+). The crossover function is defined by the following connections between MAU A and MAU B shown in Table 3.

TABLE 3. MAU A AND MAU B CONNECTIONS

Pin	MAU A	MAU B	
Pin	Signal	Signal	Pin
1	TD+(A)	RD+(B)	3
2	TD−(A)	RD−(B)	6
3	RD+(A)	TD+(B)	1
6	RD−(A)	TD−(B)	2

When an embedded crossover function is used in a DTE-to-repeater connection the crossover is usually embedded in the repeater MAU. In general, repeater MAUs have an embedded crossover and DTE MAUs do not. With proper use of the crossover function, repeaters can be cascaded through twisted pair ports and two DTEs can be connected in a point-to-point network. Repeaters can be cascaded two ways. First, one or more twisted pair ports on a repeater can be designed to have a switched (optional) crossover function. This allows a DTE connection on that port when the crossover is active; or a repeater connection, when the crossover is disabled. Secondly, two twisted pair ports with embedded crossovers can be connected by using a third external crossover.

2.2 Integrating with Existing 802.3 Networks.

Because 10BASE-T networks are fully compatible with existing 802.3 networks at the medium access control and physical layer signaling sublayers, 10BASE-T networks can be integrated with existing 802.3 networks to form one large network. The IEEE standard for repeaters allows connecting different wire types in 10-Mb/s baseband networks. This is because the repeater definition stops at the AUI connection (DTE sex). The wire type is determined by the MAU attached to the AUI connector, and can vary from port to port. Optionally, a repeater can have embedded MAUs on any of its ports. The only requirement for an embedded MAU is that functionality at the medium dependent interface point (e.g., coax tap or twisted pair connector) be maintained as if the MAU were external.

The 82505TA Multiport Repeater Controller provides embedded MAUs on 11 of its 12 ports, and an AUI connection on the remaining port. This allows creating local twisted pair subnetworks connected to an Ethernet backbone. Care must be taken not to violate the following system topology rules of 802.3 networks.

- Only one active signal path is allowed between any two stations on the network.
- No more than four repeaters are allowed in the signal path between any two stations on the network.
- There is an overall limit of 1024 stations on a network (repeaters do not count as stations).

2.3 Software Compatibility

Because the 10BASE-T definition is restricted to the MAU, software is not affected. Twisted pair networks use the same LAN controller chips (82586, 82590, 82592, and the 82596 family) as current Ethernet and Cheapernet networks and are fully software compatible.

3.0 NETWORK SYSTEM COMPONENT DESIGN

The design of various 10BASE-T network system components is presented in this section.. First, DTEs with embedded MAUs, then external MAUs, and lastly repeater designs.

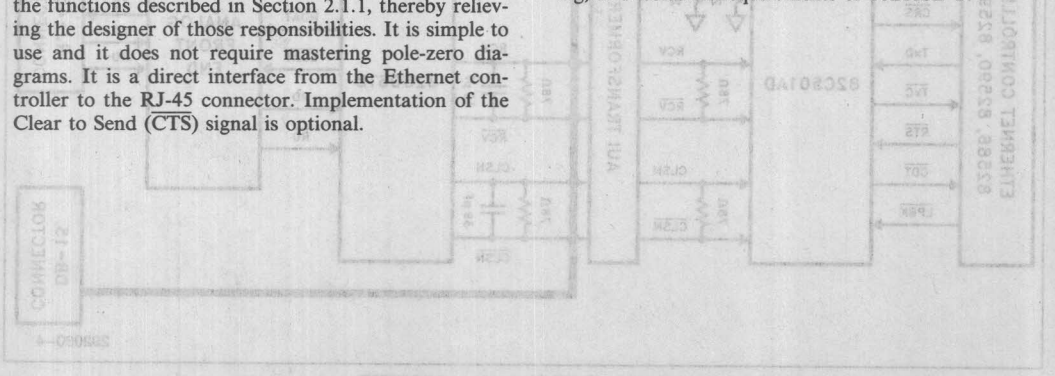
3.1 Designing a DTE Node Based on the 82521TB Serial Supercomponent.

A design for an 82521TB based DTE node with an embedded MAU is shown in Figure 3. It includes all of the functions described in Section 2.1.1, thereby relieving the designer of those responsibilities. It is simple to use and it does not require mastering pole-zero diagrams. It is a direct interface from the Ethernet controller to the RJ-45 connector. Implementation of the Clear to Send ($\overline{CTS}$) signal is optional.

The layout of the 82521TB and the RJ-45 connector should keep the TD+, TD−, RD+, and RD− signal lines as short as possible. The width of these signals should be at least twice the spacing between the signal trace layer and the ground plane. The power supply traces (V_{CC}, V_{EE}, and ground) should be as thick as possible, and bypass capacitors should be placed between each power supply and ground. An alternating strategy of 0.1- and 0.001-μF decoupling capacitors should be used throughout the host circuit board. We also recommend laying out the 82521TB on a ground plane and connecting logic ground to chassis ground.

3.2 Designing a DTE Node Based on the 82506TB Twisted Pair MAU Chip

Figure 4 shows a DTE node with an embedded MAU based on the 82506TB. It shows the Ethernet LAN controller, the 82C501AD, an AUI transformer, the 82506TB, the analog front-end, and the connector. As in previous Ethernet designs, the LAN controller provides the MAC services such as transmission deferral, collision backoff and retransmission, CRC generation and checking, and address checking. It also provides the host interface. The 82C501AD provides the serial interface function of Manchester encoding and decoding and clock recovery. The 82506TB provides the MAU functions, which include carrier sense, collision detect link integrity, jabber protection, twisted pair line driving, and line receiving. The analog front-end handles the preconditioning summation, filtering, balancing, and isolation requirements of 10BASE-T.



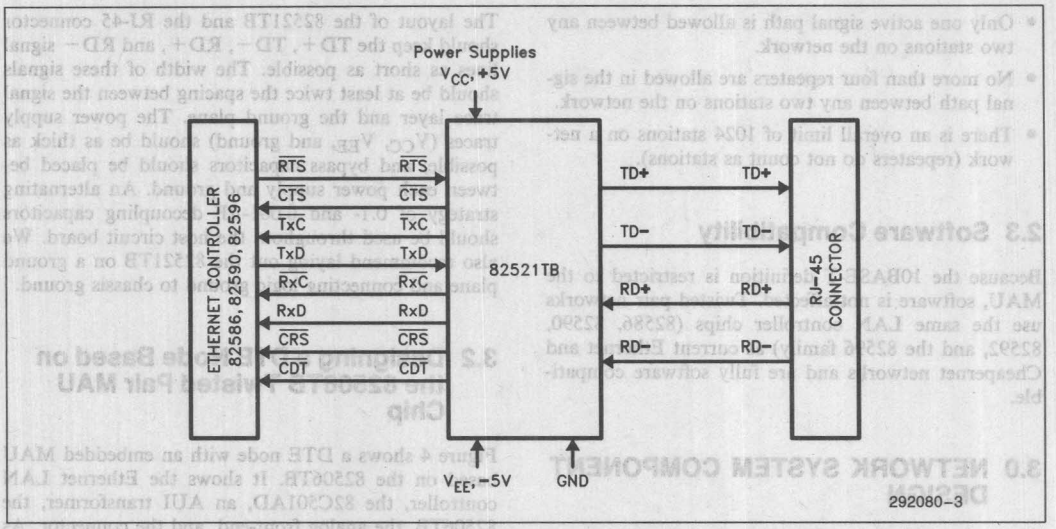


Figure 3. 82521TB SSC Interface

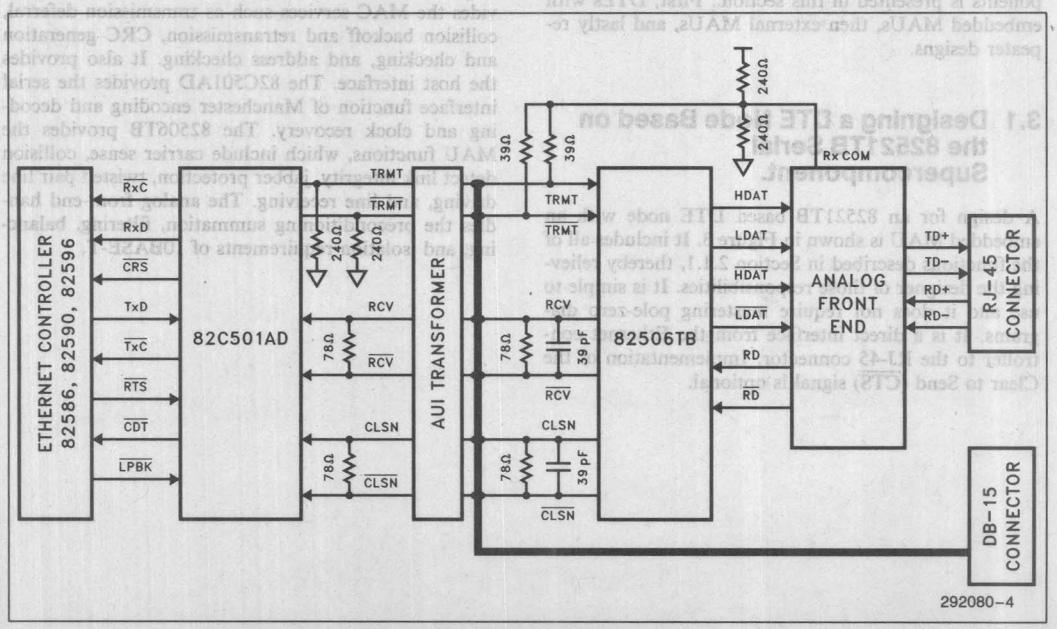


Figure 4. DTE Based on 82506TB

3.2.1 HOST TO ETHERNET LAN CONTROLLER INTERFACE

The 82586 Ethernet LAN Controller interface is included in the Intel application note, AP-274.

3.2.2 THE ETHERNET LAN CONTROLLER TO 82C501AD INTERFACE

The 82C501AD to Ethernet controller interface consists of the direct connection of $\overline{TxC}$, TxD, $\overline{RxC}$, RxD, $\overline{RTS}$ ($\overline{TEN}$), $\overline{CRS}$, and $\overline{CDT}$ signals to the controller.

A 20-MHz ± 0.01% crystal, or crystal oscillator, is recommended for clocking the 82C501AD. Many crystals that meet the requirements of the 82C501AD are available commercially and are listed in the 82C501AD data sheet.

3.2.3 THE 82C501AD TO 82506TB INTERFACE

The 82C501AD to 82506TB interface is a standard AUI. It includes an isolation transformer to provide dc common mode isolation.

3.2.4 THE 82506TB TO ANALOG FRONT-END INTERFACE (AFE)

The four transmit outputs of the 82506TB (HDAT, LDAT, $\overline{HDAT}$, and $\overline{LDAT}$) and two receive inputs (RD and $\overline{RD}$) are connected to the analog front- end described in Section 4. The analog front- end handles preconditioning voltage summation, EMI filtering, and isolation.

3.3 Using the 82523TB Twisted Pair Ethernet MAU to Convert Existing Ethernet Nodes to 10BASE-T

Any Ethernet node that supports the AUI (DB-15 connector) can be converted to a 10BASE-T node by using the 82523TB MAU. A standard AUI cable is used to connect the DB-15 connectors on the node and the 82523TB. The 10BASE-T cable then plugs into the RJ-45 receptacle and the node is connected. The interface is shown in Figure 5.

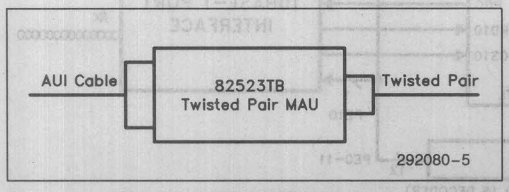


Figure 5. 82523TB Interface

3.4 Designing a MAU Using the 82506TB Twisted Pair MAU Chip

Figure 6 shows a stand-alone MAU design using the 82506TB. It contains the same AUI transformer, 82506TB, analog front-end, and connector as the embedded design. The serial interface functions and MAC services are contained in a separate DTE. The stand-alone MAU functions include carrier sense, collision detection, link integrity, jabber protection, twisted pair line driving, and line receiving. The analog front-end handles the preconditioning summation, filtering, balancing, and isolation requirements of 10BASE-T.

3.5 Designing a Multiple Port Repeater Based on the 82505TA MPR

Figure 7 shows an MPR based on the 82505TA (with one 82504TA). It contains 11 twisted pair ports with embedded MAUs and 1 AUI port. The 82505TA controls the operation of the repeater in accordance with ANSI/IEEE 802.3c-1988 repeater unit specifications, which include signal retiming, automatic preamble generation, autopartitioning, and jam signal generation. The 82504TA does Manchester decoding and clock recovery while an incoming signal is active. Two addressable latches (74LS259s) are used to control the 16 LED indicators. A 4-to-16 decoder (74LS154) is used to disable the transmitter of the receiving port during transmission without contention. The twisted pair port functions contain the line drivers, the line receivers, the filter, and the isolation required for a twisted pair embedded MAU. In addition, one AUI interface is present to provide access to existing (IEEE 802.3) 10-Mb/s baseband segments.

3.5.1 82505TA TO 82504TA INTERFACE AND CLOCK GENERATION

The 82505TA to 82504TA interface, shown in Figure 4, is straightforward. It consists of six signals directly connected between the devices. The signals are TRxD, TPS, $\overline{MCV}$, $\overline{CRS}$, $\overline{RxC}$, and RxD. The 82504TA performs the Manchester decoding and clock recovery for the repeater.

A single clock oscillator is recommended for clocking the 82505TA and 82504TA. The requirements are identical to those shown for the DTE design using the 82504TA.

- Frequency Tolerance. ≤0.01%
- Rise and Fall Times. ≤5 ns
- Duty cycle. 60/40% or better
- Output. TTL compatible

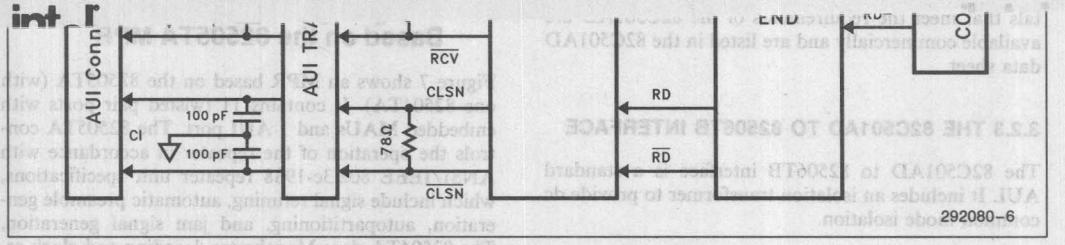


Figure 6. MAU Based on 82506TB

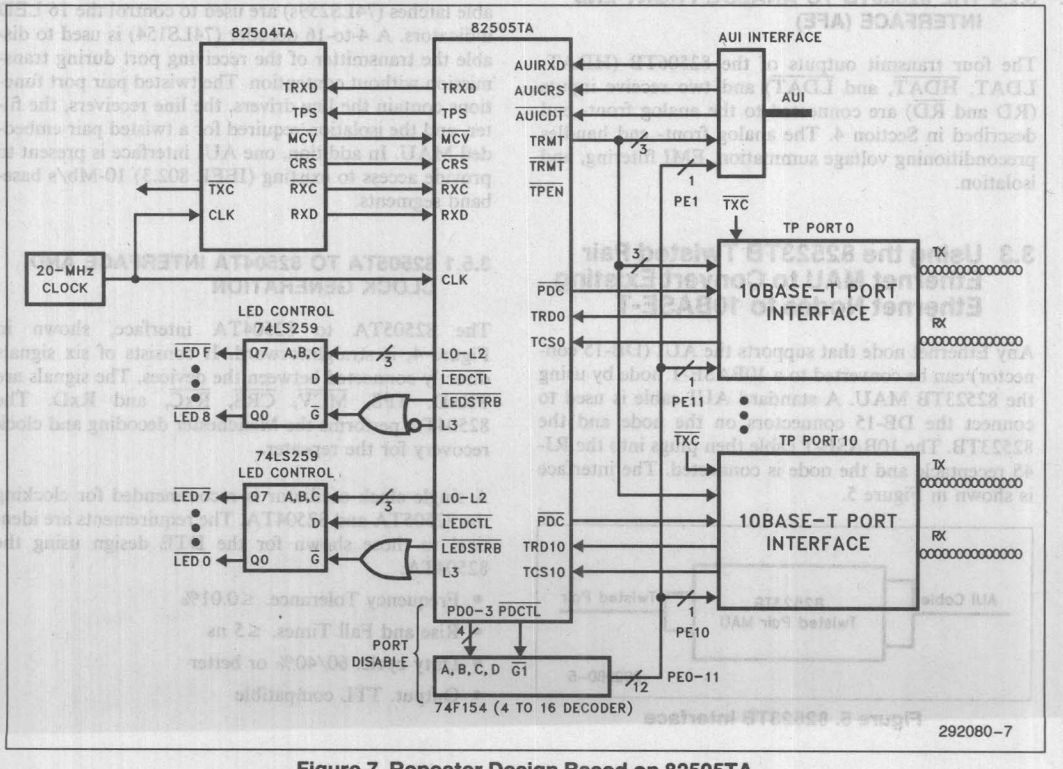


Figure 7. Repeater Design Based on 82505TA

3.5.2 TWISTED PAIR PORT DESIGN

The twisted pair port circuit, shown in Figures 8 and 9, provides the 10BASE-T functions for each twisted pair port. It uses the same AFE (see Section 4) as the 82506TB designs.

82505TA to AFE Transmit Interface. This interface converts the four transmit signals (TRMT, $\overline{\text{TRMT}}$, $\overline{\text{PDC}}$, and $\overline{\text{TPEN}}$) from the 82505TA, and the port enable (PEx) signal from the port disable control, to the four transmit signals of the analog front-end (TDH, TDL, $\overline{\text{TDH}}$, and $\overline{\text{TDL}}$). The design shown here uses an octal line driver (74ACT244) with the drivers paired.

The circuit operates as follows. The TRMT and $\overline{\text{TRMT}}$ signal are XOR'd with the $\overline{\text{PDC}}$ signal to generate the proper logic signals for preconditioning. These signals are then gated by TXE (Transmit enable). a function of the $\overline{\text{TPEN}}$ signal and the PEx (Port Enable) signal. The signals are blocked whenever the repeater is idle, the Port Enable for this port is false, or the 10BASE-T state machines have disabled transmission. The gated signals then pass through the link beat generator. Link beats are broadcast during long periods of idle. The signals now pass to the 74ACT244 line drivers, through the AFE, and onto the twisted pair.

Line Receivers. The incoming receive signal passes through the AFE into a gated line receiver controlled by the squelch circuitry. The line receiver converts the received differential signal to TTL levels and feeds it to the MPR. The receiver can be designed with a zero crossing detector (e.g., NE521) and is gated with the TCSx signal.

State Machines. The state machines are required by the 10BASE-T draft standard; they handle link integrity and jabber protection. The implementation shown uses an Intel 5C180 EPLD. The equations are in iPLSII format and are in the appendix.

Squelch Circuit. The squelch circuit distinguishes noise from valid link beats and incoming data on the receive pair by detecting signals above a preset voltage level. When there is no signal on the receive pair the squelch circuit deasserts the CARR and LINK signals. Link is asserted when a signal above the threshold arrives. CARR is asserted if the required number of bits are seen. This causes TCSx to assert and the line receiver to be enabled.

3.5.3 AUI PORT

The AUI port circuitry is shown in Figure 10. It comprises interface circuits, the DO line drivers, two quad D-flip-flops (74F175s), and terminated line receivers for the DI (squelch and data) and CI (squelch only) circuits.

The CI squelch line receiver feeds the D-0 and clear inputs for one of the quad D-flip-flop circuits. When a signal greater than the squelch offset is seen, the flip-flops are cleared and $\overline{\text{AUICDT}}$ is asserted. This continues as long as CI is active. S quelch receiver output is held high during the start of idle and the flip-flops set in sequence. After four clocks, 150 to 200 ns, the last flip-flop is set and $\overline{\text{AUICDT}}$ deasserts. It remains deasserted during the entire idle period.

The DI line receivers work in much the same way, except that activity on CI, or an active transmission, will inhibit AUICRS. The data channel on DI is processed without a voltage offset and is gated by AUICRS. This way, the least amount of jitter is added on the AUIRxD line and the data channel is not sensitive to idle noise.

The DO line drivers are controlled by $\overline{\text{TPEN}}$ and PE11; the drivers should activate when both are asserted. A voltage divider is provided after the drivers to achieve the proper driver levels.

3.5.4 PORT DISABLE CONTROL

Port disable control is handled by a 74LS154 4-to-16 decoder. During transmission without contention the address of the originating port is given to the decoder and the control line asserted. This disables the transmitter to that port. When a transmit based collision occurs the control line to the decoder is deasserted and jam is broadcast on all ports.

3.5.5 LED CONTROL

Two 8-bit, addressable latches (74LS259s) handle this function. The controller cycles through the addresses for the LEDs each 105 ms and will turn each one on or off. The three least significant address bits (L_2–L_0) for the LED control are fed to each 8-bit latch. The most significant address bit (L_3) controls the enable line to the two packages. The $\overline{\text{LEDCTRL}}$ signal determines the state of the LED when it is strobed by $\overline{\text{LEDSTRB}}$.

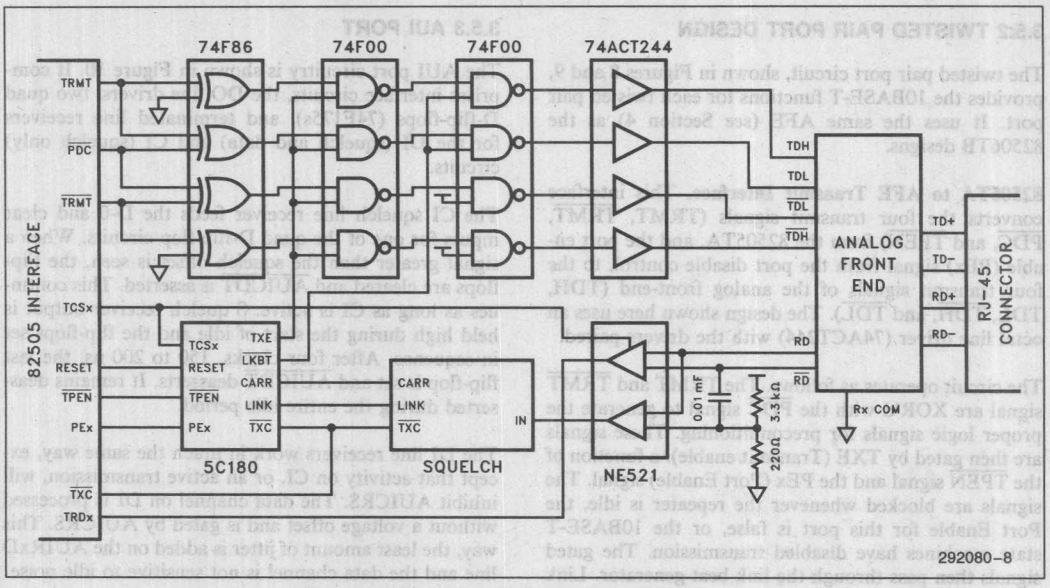


Figure 8. Repeater 10BASE-T Interface Port

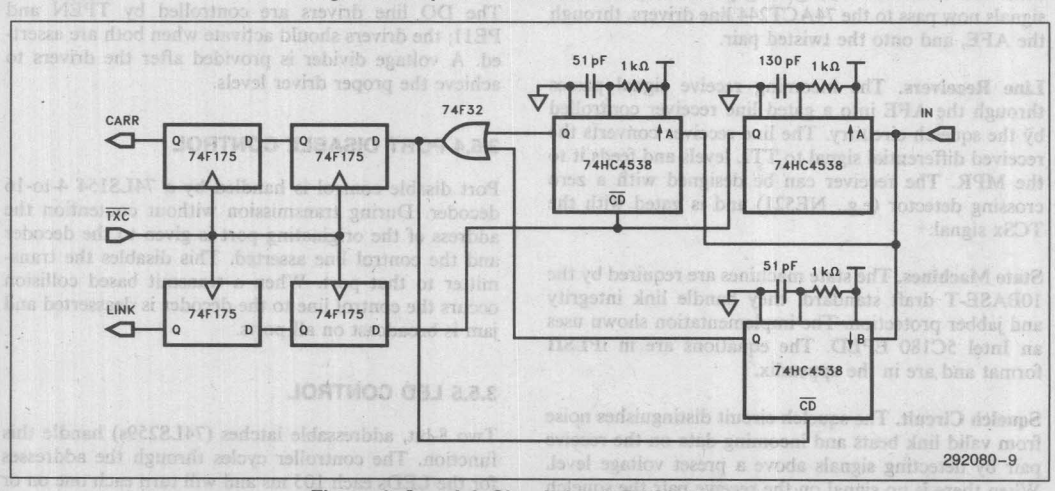


Figure 9. Squelch Circuit for 10BASE-T Port

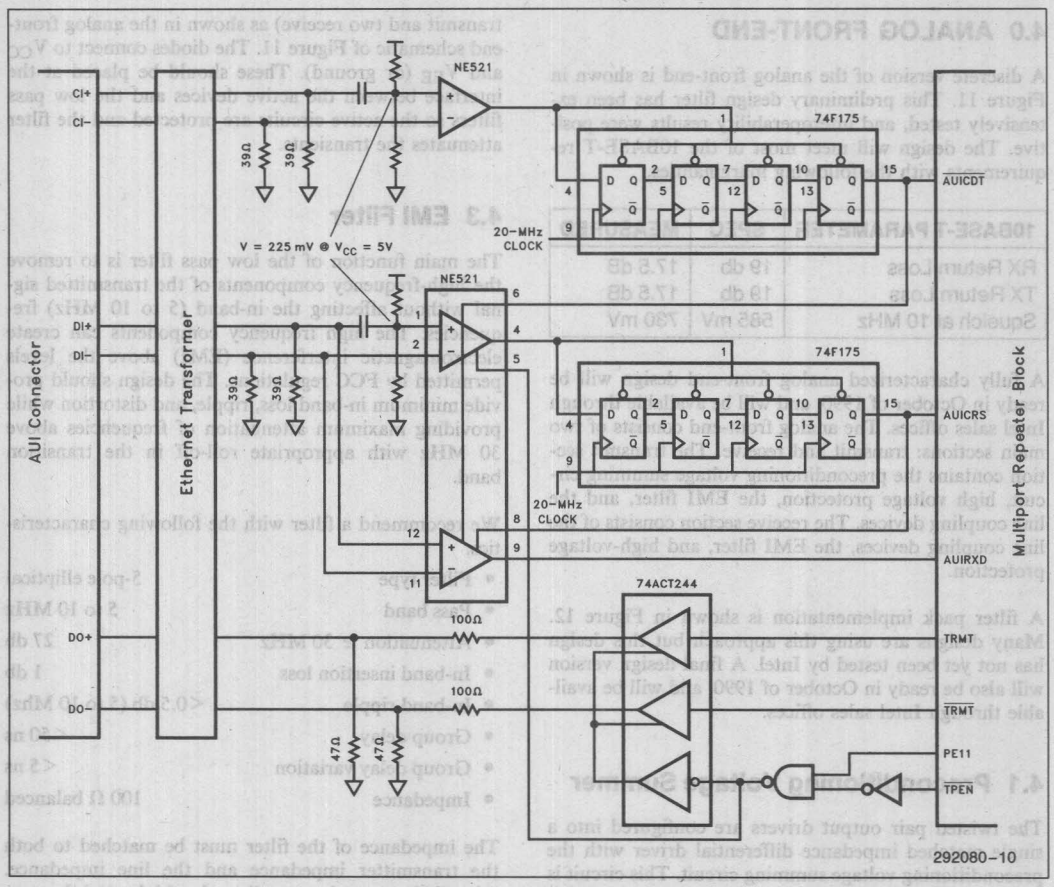


Figure 10. AUI Port

line coupling devices, the EMI filter, and high-voltage protection.

A filter pack implementation is shown in Figure 12. Many designs are using this approach but this design has not yet been tested by Intel. A final design version will also be ready in October of 1990, and will be available through Intel sales offices.

4.1 Preconditioning Voltage Summer

The twisted pair output drivers are configured into a single matched impedance differential driver with the preconditioning voltage summing circuit. This circuit is designed to give a preconditioned differential signal. During "thin" pulses, and the first half of "fat" pulses, the differential driver provides 100% drive level power. During the second half of "fat" pulses it provides only 33% drive level as required by the preconditioning algorithm.

This circuit provides a constant source impedance whenever power is applied; this controls matching the driver impedance to the twisted pair cable impedance. This will limit reflections that would result in excessive noise.

4.2 High-Voltage Protection

Protection should be provided to prevent the active devices being damaged by high-voltage transients from the twisted pair line. We recommend placing a pair of diodes on each of the four differential signals (two

We recommend a filter with the following characteristics.

- Filter type: 5-pole elliptical
- Pass band: 5 to 10 MHz
- Attenuation ≥ 30 MHz: 27 db
- In-band insertion loss: 1 db
- In-band ripple: <0.5 db (5 to 10 Mhz)
- Group delay: <50 ns
- Group delay variation: <5 ns
- Impedance: 100 Ω balanced

The impedance of the filter must be matched to both the transmitter impedance and the line impedance. Also, balance and grounding should be tightly controlled for proper operation. Due to these considerations we recommend a differential filter built symmetrically on each line of the differential pairs with the impedance matched at each end.

4.4 Line Coupling Devices

The line coupling devices include the transformers and common mode choke. The transformers provide ac coupling between the line and the circuitry while providing dc isolation. The recommended minimum isolation is 2250 V_{DC}. The windings should be identical to provide proper balance between the two ends of the transformers. To provide appropriate impedance matching in the frequency range of interest, the transformers should have appropriate primary and secondary inductance (200 μH typical) and minimal interwinding capacitance (<20 pF).

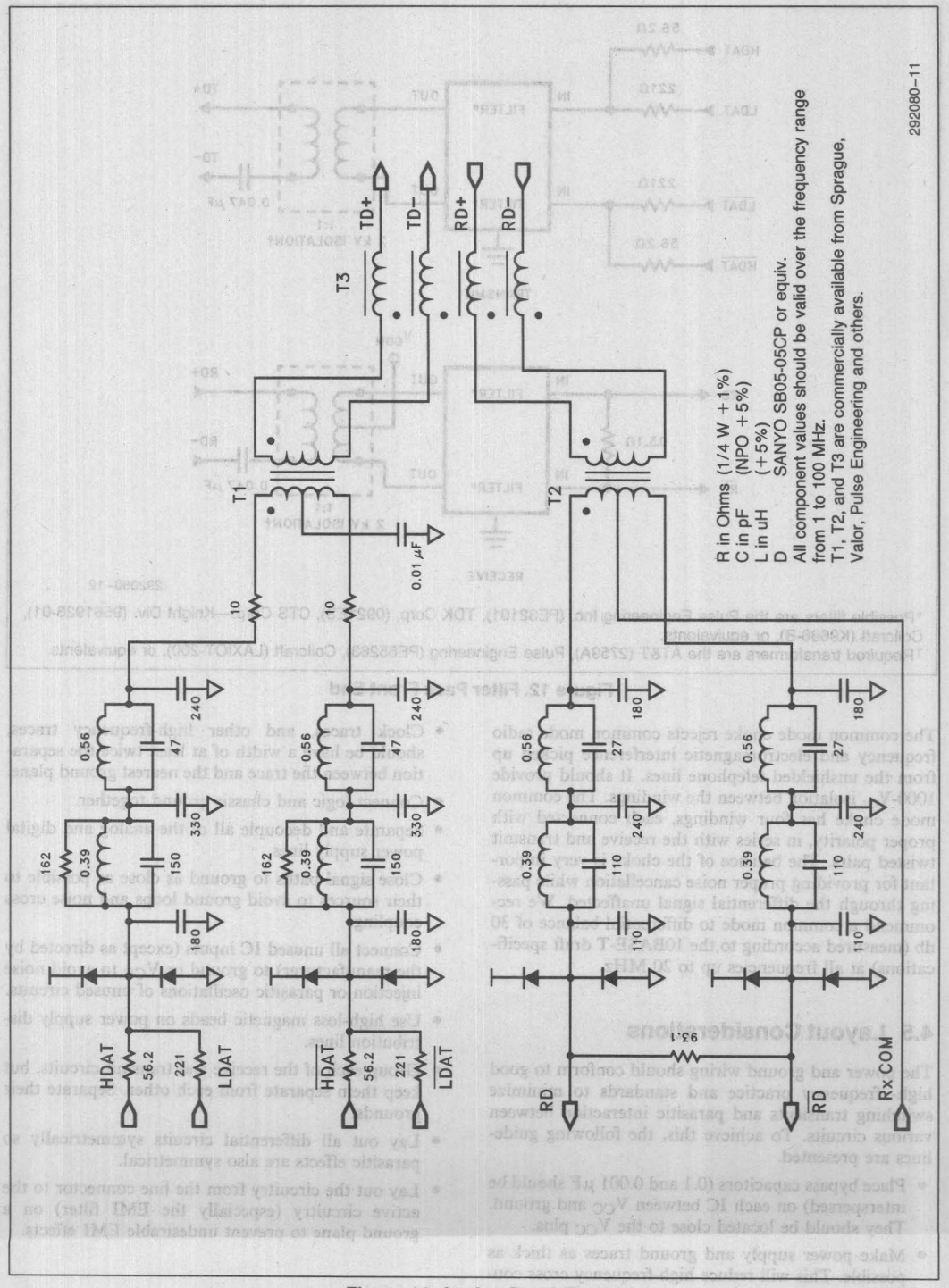
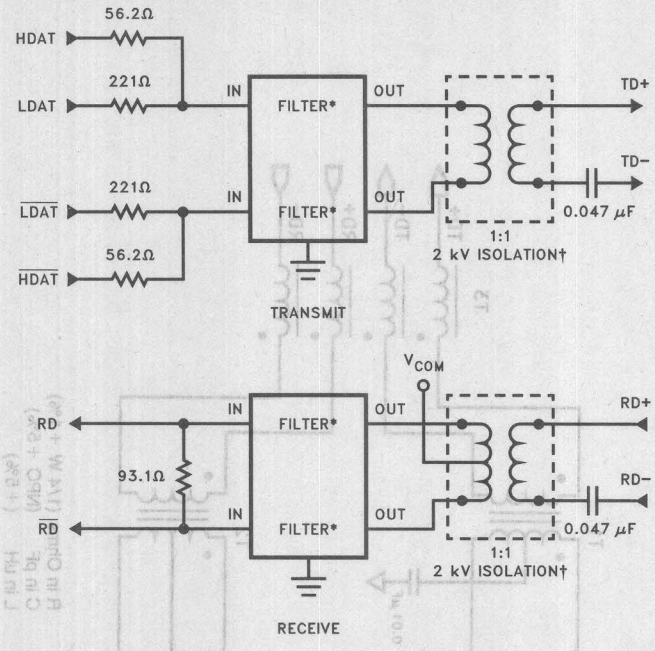


Figure 11. Analog Front-End



*Possible filters are the Pulse Engineering Inc. (PE32101), TDK Corp. (0921ES), CTS Corp.—Knight Div. (9561928-01), Coilcraft (K9686-B), or equivalents.
†Required transformers are the AT&T (2759A), Pulse Engineering (PE65263), Coilcraft (LAXIOT-200), or equivalents.

Figure 12. Filter Pack Front End

The common mode choke rejects common mode radio frequency and electromagnetic interference picked up from the unshielded telephone lines. It should provide 1000-V_{dc} isolation between the windings. The common mode choke has four windings, each connected with proper polarity, in series with the receive and transmit twisted pairs. The balance of the choke is very important for providing proper noise cancellation while passing through the differential signal unaffected. We recommend a common mode to differential balance of 30 db (measured according to the 10BASE-T draft specifications) at all frequencies up to 20 MHz.

4.5 Layout Considerations

The power and ground wiring should conform to good high- frequency practice and standards to minimize switching transients and parasitic interaction between various circuits. To achieve this, the following guidelines are presented.

- Place bypass capacitors (0.1 and 0.001 μF should be interspersed) on each IC between V_{CC} and ground. They should be located close to the V_{CC} pins.
- Make power supply and ground traces as thick as possible. This will reduce high-frequency cross coupling caused by the inductance of thin traces.
- Clock traces, and other high-frequency traces, should be have a width of at least twice the separation between the trace and the nearest ground plane.
- Connect logic and chassis ground together.
- Separate and decouple all of the analog and digital power supply lines.
- Close signal paths to ground as close as possible to their sources to avoid ground loops and noise cross coupling.
- Connect all unused IC inputs (except as directed by the manufacturer) to ground or V_{CC} to avoid noise injection or parasitic oscillations of unused circuits.
- Use high-loss magnetic beads on power supply distribution lines.
- Group each of the receive and transmit circuits, but keep them separate from each other. Separate their grounds.
- Lay out all differential circuits symmetrically so parasitic effects are also symmetrical.
- Lay out the circuitry from the line connector to the active circuitry (especially the EMI filter) on a ground plane to prevent undesirable EMI effects.

5.0 SUMMARY

This application note presented several designs meeting the IEEE 802.3 10BASE-T draft standard 10BASE-T. They use standard telephone twisted pair wiring and a star configuration for cost savings and flexibility. They use the same IEEE 802.3 standard for CSMA/CD medium access control and, where applicable, the physical layer signaling. This network type is fully software compatible with, and can connect to present Ethernet or Cheapernet networks. The hardware connection is made through an 802.3 defined AUI port and by complying with the repeater standard ANSI/IEEE 802.3c-1988.

Intel has introduced four products for designing network components (DTEs and repeaters). DTE design can be done with either the 82521TA Serial Supercomponent or the 82506TB TP Transceiver Chip. The supercomponent contains all the circuitry required between the Ethernet controller and the RJ-45 connector. DTEs that support the AUI can be instantly connected to 10BASE-T networks using the 82523TB MAU supercomponent. Multiple port repeaters can be designed using the 82505TA with an 82504TA. It allows for 11 twisted pair ports and 1 AUI port.

1

```
Feb. 20, 1990
Repeater 180
Rev 0
5C180
5C180 for Multiple Port Repeater-  Counters, State machines, etc.
OPTIONS: TURBO=ON

PART:  5C180

INPUTS:  TXC#, LID, WDTD, TPEN#, PEx,
         TEST, CARR, LINK, RESET

OUTPUTS:  TCSx, LKBT#, TXE, WDT,
          SLOCLK, LI

          JT14_RESET, JT58_RESET, LT_RESET, SCQ01, SCQ02,
          SCQ03, SCQ04,
          SCQ05, SCQ06, SCQ07, SCQ08, SCQ09, SCQ10,
          SCQ11, SCQ12, SCQ13, DLINK, JTQ1, JTQ2,
          JTQ3, JTQ4, JTQ5, JTQ6, JTQ7, JTQ8,
          LTQ1, LTQ2, LTQ3, LTQ4, LTQ5, LCQ1, LCQ2,
          JFQ1, LIFQ1, LIFQ2

NETWORK:
          TXCn     = INP(TXC#)
          LID      = INP(LID)
          WDTD     = INP(WDTD)
          TPENn    = INP(TPEN#)
          TEST     = INP(TEST)
          CARRIER  = INP(CARR)
          SLINK    = INP(LINK)
          RESET    = INP(RESET)
          PEx      = INP(PEx)

          TCSx     = CONF(TCSx,VCC)
          LKBT#    = RONF(LBEATn,TXCn,GND,GND,VCC)
          LKBTI    = NORF(ILBEAT,TXCn,GND,GND)
          TXE      = CONF(TXE,VCC)
```

292080-13

```
% Timer reset feedback macrocells %
JT14_RESET = NOCF(JT14_RESETd)
JT58_RESET = NOCF(JT58_RESETd)
LT_RESET   = NORF(LT_RESETd,TXCn,GND,GND)

% SLOCLK macrocells %
SCQ01 = NOTF(SCT01,TXCn,RESET,GND)
SCQ02 = NOTF(SCT02,TXCn,RESET,GND)
SCQ03 = NOTF(SCT03,TXCn,RESET,GND)
SCQ04 = NOTF(SCT04,TXCn,RESET,GND)
SCQ05 = NOTF(SCT05,TXCn,RESET,GND)
SCQ06 = NOTF(SCT06,TXCn,RESET,GND)
SCQ07 = NOTF(SCT07,TXCn,RESET,GND)
SCQ08 = NOTF(SCT08,TXCn,RESET,GND)
SCQ09 = NOTF(SCT09,TXCn,RESET,GND)
SCQ10 = NOTF(SCT10,TXCn,RESET,GND)
SCQ11 = NOTF(SCT11,TXCn,RESET,GND)
SCQ12 = NOTF(SCT12,TXCn,RESET,GND)
SCQ13 = NOTF(SCT13,TXCn,RESET,GND)
SLOCLK,SCQ14 = TOTF(SCT14,TXCn,RESET,GND,VCC)
SLOCK = CLKB(SCQ14)
DLINK = NORF(SLINK,TXCn,RESET,GND)

% Jabber timer macrocells %
JTQ1 = NOTF(JTT1,SLOCK,JT14_RESET,GND)
JTQ2 = NOTF(JTT2,SLOCK,JT14_RESET,GND)
JTQ3 = NOTF(JTT3,SLOCK,JT14_RESET,GND)
JTQ4 = NOTF(JTT4,SLOCK,JT58_RESET,GND)
JTQ5 = NOTF(JTT5,SLOCK,JT58_RESET,GND)
JTQ6 = NOTF(JTT6,SLOCK,JT58_RESET,GND)
JTQ7 = NOTF(JTT7,SLOCK,JT58_RESET,GND)
JTQ8 = NOTF(JTT8,SLOCK,JT58_RESET,GND)


% Link test timer macrocells %
LTQ1 = NOTF(LTT1,SLOCK,LT_RESET,GND)
LTQ2 = NOTF(LTT2,SLOCK,LT_RESET,GND)
LTQ3 = NOTF(LTT3,SLOCK,LT_RESET,GND)
LTQ4 = NOTF(LTT4,SLOCK,LT_RESET,GND)
LTQ5 = NOTF(LTT5,SLOCK,LT_RESET,GND)


% Link count macrocells %
LCQ1 = NOTF(LCT1,TXCn,RESET,GND)
LCQ2 = NOTF(LCT2,TXCn,RESET,GND)
```

292080-14

```
EQUATIONS:
      % Input controlled variables %
          output_active = !(TPENn + !PEx);
          output_idle = TPENn + !PEx;
          input_active = CARRIER;
          input_idle = !CARRIER;
          link_test_rcvd = SLINK*!DLINK;

      % Output equations %
          TCSx = input_active * !disable_receiver;
          ILBEAT = linkbeat_timer_done;
          LBEATn = !(ILBEAT * !LKBTI);

          TXE = !disable_driver * output_active;
          disable_driver = disable_driver_3 + disable_driver_4;

      % Miscellaneous equations %
          link_count_is_3 = LCQ2 * LCQ1;
          LCRESET = LINK_TEST_FAIL_RESET * !link_test_rcvd;
          LCCOUNT = LINK_TEST_FAIL_RESET * link_test_rcvd +
                                          LINK_TEST_FAIL  *  link_test_rcvd  *
link_test_min_timer_done;
          LCT1 = LCCOUNT + LCRESET * LCQ1;
          LCT2 = LCCOUNT * LCQ1 + LCRESET * LCQ2;
% ------------------------------------------------------- %

    % SLOCLK counter equations %
       SCT01 = VCC;
       SCT02 = SCQ01;
       SCT03 = SCT02 * SCQ02;
       SCT04 = SCT03 * SCQ03;
       SCT05 = SCT04 * SCQ04;
       SCT06 = SCT05 * SCQ05;
       SCT07 = SCT06 * SCQ06;
       SCT08 = SCT07 * SCQ07;
       SCT09 = SCT08 * SCQ08;
       SCT10 = SCT09 * SCQ09;
       SCT11 = SCT10 * SCQ10;
       SCT12 = TEST + !TEST * SCT11 * SCQ11;
       SCT13 = SCT12 * SCQ12;
       SCT14 = SCT13 * SCQ13;

      % JABBER timer equations %
          transmit_timer_done = JTQ5 * NON_JABBER_OUTPUT;
          unjab_timer_done = JTQ8 * JTQ7 * UNJAB_WAIT;
          unjab_timer_not_done = !unjab_timer_done;
          linkbeat_timer_done = JTQ3 * !NON_JABBER_OUTPUT * !LID;
          JT14_RESETd = RESET + NO_OUTPUT * output_active
             + NON_JABBER_OUTPUT * output_idle;
          JT58_RESETd = JT14_RESETd
                      + JAB * output_idle;
```

292080-15

```
    JTT1 = VCC;
    JTT2 = JTT1 * JTQ1;
    JTT3 = JTT2 * JTQ2;
    JTT4 = JTT3 * JTQ3;
    JTT5 = JTT4 * JTQ4 * JTCOUNT;
    JTT6 = JTT5 * JTQ5;
    JTT7 = JTT6 * JTQ6;
    JTT8 = JTT7 * JTQ7;
    JTCOUNT = !(JTQ5 * JTQ6 * JTQ7 * JTQ8);


% LINK timer equations %
    link_loss_timer_done =LTQ5 *LTQ4 *LTQ3 *LTQ2 *LTQ1 *IDLE_TEST;
    link_test_min_timer_done =
          (LTQ3 + LTQ4 + LTQ5) * ((IDLE_TEST) + (LINK_TEST_FAIL));
    link_test_min_timer_not_done = !link_test_min_timer_done;
    link_test_max_timer_done = LTQ5 * LINK_TEST_FAIL;
    LT_RESETd = IDLE_TEST * (input_active
              + (link_test_rcvd * link_test_min_timer_done))
             + LINK_TEST_FAIL_RESET
             + LINK_TEST_FAIL * link_test_min_timer_done
               * link_test_rcvd
             + LINK_TEST_FAIL_EXTEND;
    LTT1 = LTCOUNT;
    LTT2 = LTCOUNT * LTQ1;
    LTT3 = LTCOUNT * LTQ1 * LTQ2;
    LTT4 = LTCOUNT * LTQ1 * LTQ2 * LTQ3;
    LTT5 = LTCOUNT * LTQ1 * LTQ2 * LTQ3 * LTQ4;
    LTCOUNT = !(LTQ1 * LTQ2 * LTQ3 * LTQ4 * LTQ5);
```

292080-16

```
% ------------------------------------------------------ %
MACHINE: JABBER_FUNCTION
CLOCK:   TXCn
CLEAR:   RESET

          STATES:  [ WDT JFQ1 ]      % JFQ2 is WDT also %
        NO_OUTPUT  [  0    0  ]
NON_JABBER_OUTPUT  [  0    1  ]
              JAB  [  1    1  ]
       UNJAB_WAIT  [  1    0  ]


  NO_OUTPUT:  IF output_active              THEN NON_JABBER_OUTPUT

  NON_JABBER_OUTPUT:  % start_transmit_timer %
                      IF (output_active * transmit_timer_done * !WDTD)
                                            THEN JAB
                      IF output_idle        THEN NO_OUTPUT

  JAB: IF output_idle                       THEN UNJAB_WAIT
       IF output_active * WDTD              THEN NO_OUTPUT
       ASSERT: disable_driver_3             % ASSERT WDT here %

  UNJAB_WAIT:    % start_unjab_timer %
                 IF output_active * unjab_timer_not_done * !WDTD
                                            THEN JAB
                 IF unjab_timer_done + WDTD THEN NO_OUTPUT
                 ASSERT: disable_driver_3   % ASSERT WDT here %
```

292080-17

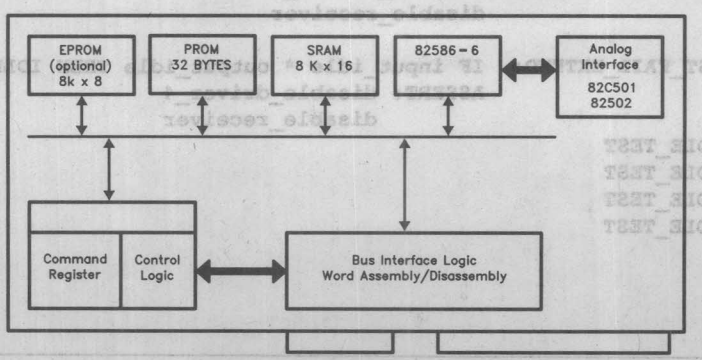
```
% ------------------------------------------------------ %
MACHINE: LINK_INTEGRITY_FUNCTION
CLOCK:   TXCn
CLEAR:   RESET

                STATES:  [  LI   LIFQ1   LIFQ2 ]
              IDLE_TEST  [  0      0       0   ]
  LINK_TEST_FAIL_RESET   [  1      1       0   ]
       LINK_TEST_FAIL    [  1      0       0   ]
 LINK_TEST_FAIL_EXTEND   [  0      0       1   ]
                 ALIA    [  0      1       1   ]
                 ALIB    [  0      1       0   ]
                 ALIC    [  1      0       1   ]
                 ALID    [  1      1       1   ]

IDLE_TEST:     %  start_link_loss_timer  %
               %  start_link_test_min_timer  %
              IF link_loss_timer_done * !LID  THEN LINK_TEST_FAIL_RESET

LINK_TEST_FAIL_RESET:  IF link_test_rcvd * !LID * input_idle
                         THEN LINK_TEST_FAIL
                       IF input_active + LID THEN LINK_TEST_FAIL_EXTEND
                       ASSERT: disable_receiver
                               disable_driver_4

 LINK_TEST_FAIL:  %  start_link_test_min_timer  %
                  %  start_link_test_max_timer  %
                  IF input_active + link_count_is_3 + LID
                    THEN LINK_TEST_FAIL_EXTEND
                  IF (link_test_max_timer_done
                     + (link_test_min_timer_not_done
                    * link_test_rcvd)) * !LID * input_idle
                                                THEN LINK_TEST_FAIL_RESET
                  ASSERT: disable_driver_4
                          disable_receiver

 LINK_TEST_FAIL_EXTEND:  IF input_idle * output_idle THEN IDLE_TEST
                         ASSERT: disable_driver_4
                                 disable_receiver

ALIA:   IDLE_TEST
ALIB:   IDLE_TEST
ALIC:   IDLE_TEST
ALID:   IDLE_TEST


END$
```

292080-18

PC586E
CSMA/CD LAN EVALUATION BOARD

- **Supports Established CSMA/CD LAN Standards:**
  - **— Ethernet (IEEE 802.3 10BASE5)**
  - **— Cheapernet (IEEE 802.3 10BASE2)**
- **Interfaces to Popular IBM and IBM Compatible PC Systems:**
  - **— IBM PC, PC-XT, PC-AT (8-Bit Data Transfer)**
  - **— IBM PC-AT (16-Bit Data Transfer)**
- **Jumper Selection Offers High Degree of Flexibility in System Configuration:**
  - **— Up to 8 Address Decode Ranges**
  - **— Up to 8 Interrupt Lines**
  - **— Ethernet (IEEE 802.3 10BASE5)**
  - **— Cheapernet (IEEE 802.3 10BASE2)**
  - **— Number of Wait-States**
- **Auto-Configuring for either 8-Bit or 16-Bit Bus Systems**
- **On-Board Transceiver Provides Direct Coaxial Connection for Cost-Effective Cheapernet Applications**
- **Pipelined Access in 8-Bit Mode Increase Performance through Reduced Wait-States**
- **16 Kbytes of Shared Memory-Mapped SRAM Enables Higher Performance Network Operation**
- **Reduces Design Complexity because No I/O Address or DMA Channels Required**
- **High Efficiency Interleaved Memory Access Permits Zero Wait-State Access by Host CPU for Most Cycles**
- **8 Kbytes of "Remote Boot" EPROM (Optional) Eliminates Need for Disk Drives**
- **Provides LAN Designer with a Complete, High-Performance CSMA/CD Ethernet/Cheapernet Solution**

The PC586E evaluation board is a non-intelligent, buffered CSMA/CD LAN adapter card designed to demonstrate Intel's high-performance Ethernet/Cheapernet chip set. It provides IEEE 802.3 TYPE 10BASE5 (Ethernet) and TYPE 10BASE2 (Cheapernet or thinwire Ethernet) connections for IBM PC, PC-XT, PC-AT and compatible systems. The PC586E combines the Intel 82586 LAN Coprocessor and the Intel 82C501 Ethernet Serial Interface with an on-board Ethernet Transceiver into a total Ethernet/Cheapernet solution. The card is easily installed in either an 8-bit or 16-bit PC expansion slot and then automatically configures itself for 8-bit or 16-bit data transfers. Its jumpers offer a high degree of flexibility for system-dependent configuration. For Ethernet applications, the 82586/82C501 pair provide the complete transceiver cable interface required by the IEEE 802.3 standard. In addition, the PC586E's on-board transceiver provides the entire coaxial cable interface for convenient, cost-effective Cheapernet systems.



290196–1

Figure 1. PC586E Block Diagram

The PC586E is provided solely as an evaluation tool for use in designing with Intel's 82586 chip set. It has not been tested for compliance to FCC requirements for EMI (Part 15, subpart j). Intel is not responsible for any misuse of this evaluation board.

October 1988
Order Number: 290196-001

The PC586E is part of Intel's LAN Evaluation Board program. The board is intended to demonstrate the high-performance characteristics of the 82586 chip set in an adapter card application. The PC586E gives LAN engineers a head start in finding the best solution for their specific network problem. PC586E boards are shipped with detailed design documentation (artwork and PAL equations also available).

The PC586E is based on an Interleaved Local Memory Access scheme with Static RAM dual-ported between the 82586 LAN Coprocessor and the Host System CPU. Access to the board is purely Memory-Mapped, and therefore, no I/O ports or DMA channels are required. In addition to the shared SRAM, the system supports a "Remote-Boot" EPROM and 32 bytes of Address PROM. The 82586 has access only to the Static RAM.

MEMORY

The Local Memory consists of 16 Kbytes of Static RAM, 32 bytes of Address PROM, 16 Command Registers, and up to 8 Kbytes of "Remote Boot" EPROM (Optional). All of the Local Memory is mapped into unused memory space of the Host System. Commands are issued to the PC586E by transferring the instruction to a Command Register. The Command Registers are used for issuing the Reset and Channel Attention signals to the 82586, enabling interrupts and configuring the board.

CONFIGURATION

There are up to 8 jumper-selectable locations for the Local Memory and the Command Registers (four of these locations are mapped above the 1 Mbyte boundary, FFFFh). In addition, the jumpers are used for the Interrupt Request Signal which may be assigned to any one of eight Interrupt Request lines.

The PC586E automatically detects if it is placed in an 8-bit or 16-bit expansion slot. When the PC586E is in a 16-bit slot, a Command Register is used to program the PC586E for either 8-bit or 16-bit data transfers. One of the Command Registers can also be used to disable the interrupt signal.

INTERLEAVED MEMORY ACCESS

The PC586E uses Interleaved Memory Access between the 82586 LAN Coprocessor and the Host System CPU to increase system performance. One read or write access is allowed by the Host System for every read or write access by the 82586. In this way, high utilization of local memory is achieved. The logic used is a "cycle-stealing" approach in which the 82586 is never given wait-states. This precludes the need for wait-state logic for the 82586 and allows the 82586 to run at 6 MHz.

When the 82586 is inactive, the interleaving logic becomes transparent and the Host System may access the Local Memory with no wait-states (16-bit buses only). This provides about a 15% to 20% boost in bus performance.

DESCRIPTION OF INTERLEAVE LOGIC

Since the 82586's READY and HOLD ACKNOWLEDGE signals are always active, only a simple arbiter is required. The Control Logic merely interleaves Host System accesses with 82586 accesses. When the 82586 is active, the Host System access will occur during the first half of the 82586 "read/write" cycle. When the 82586 is inactive, the Host System access will occur at the speed of the Host Bus.

If the Host System initiates access to the static RAM during T1 or T2 of the 82586 "read/write" cycle, it will complete operation without any additional wait-states. If the Host System should initiate access during T2 or T3 of the 82586 "read/write" cycle, a maximum of three wait-states will be inserted for an 8 MHz AT system. The maximum number of wait-states depends on the width and frequency of the Host System.

WORD ASSEMBLY/DISASSEMBLY

For systems with 8-bit data buses, the PC586E has a special Word Assembly/Disassembly function. Access to the Static RAM may be made either as 8-bit or 16-bit operations. If 8-bit transfers are made, the Word Assembly/Disassembly logic is used to increase performance.

WORD DISASSEMBLY

An 8-bit "read" operation to an even address causes 16 bits of data to be read from the Static RAM. The first 8 bits are transferred onto the Host bus and the second 8 bits (corresponding to the odd address) are temporarily stored in a latch. When the subsequent "read" is made to the odd address, the data stored in the latch is copied onto the Host Bus. In this way, access to the Static RAM by the Host CPU is reduced by 50%.

WORD ASSEMBLY

An 8-bit "write" operation to an even address causes the data stored at this location to be temporarily

The Word Assembly/Disassembly function is only used for access to the Static RAM. All accesses to the Address PROM, Remote Boot EPROM and Command Registers are made as 8-bit transfers only.

REMOTE BOOT EPROM

An optional 8192 byte EPROM may be installed for either "Remote Boot" operation or general purpose ROM. Upon booting the system, the Host CPU searches for a 55AAh data pattern starting at address C8000h. If the pattern is not found, additional attempts will be made at subsequent addresses in 2 Kbyte increments. If the pattern is found, the Host will then search for a jump instruction and a Cyclic Redundancy Check (CRC). If these are found, the CPU will begin executing the code at the location specified by the jump instruction. In order to take advantage of the "Remote Boot" option, the software on the EPROM must be able to configure the PC586E and copy the operating system through the network. This ability removes the need for disk drives. The EPROM may be used for general purpose storage instead of remote booting. In either case, only 8-bit "read" operations are permitted from this device.

ETHERNET/CHEAPERNET SELECTION

The PC586E Board is jumper-selectable to operate in either Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) mode.

ETHERNET

In Ethernet mode, the 82586 LAN Coprocessor is used in conjunction with the Intel 82C501 Ethernet Serial Interface. Functions of the 82C501 include plete, on-board, coaxial cable interface.

COMPONENT DESCRIPTION

82586 LAN Coprocessor

— Implements a Complete CSMA/CD Data Link

— Incorporates all Logic for Executing Time Critical Functions Independently of Host System

— High-Level Command Interface Simplifies Software Programming

— Supporting Industry CSMA/CD LAN Standards
  Ethernet (IEEE 802.3 10BASE5)
  Cheapernet (IEEE 802.3 10BASE2)

— Provides On-Chip Memory Management with Automatic Buffer Chaining and Reclaiming

— Interfaces to Industry Standard 8-Bit and 16-Bit Microprocessors

— Powerful System Interface
  On-Chip DMA Control Allows Up to
  5 Mbytes/Sec Bus Capacity
  8-Bit or 16-Bit Data Bus
  Back-to-Back Frame Reception at 10 Mb/s

— Built-In Network Management and Diagnostics
  Transmission/Reception Error Reporting
  Network Activity and Error Statistics
  Station Diagnostics (External Loopback)
  Self Test Diagnostics

The 82586 is an intelligent peripheral that completely manages the processes of transmitting and receiving frames of data over the network, thus offloading the Host CPU of communication management tasks. The 82586 features an on-chip DMA controller which allows it to access the local memory though an efficient buffer chaining mechanism. Other features of the 82586 are the ability to perform network management activities including error and collision tallies and diagnostic capabilities via the internal and external loopback function. Control of the 82586 is through high level commands such as TRANSMIT and CONFIGURE.

All information passed between the 82586 and the Host board is made through shared local memory. The Host may load the memory with a command and prompt the 82586 to execute. While receiving a packet, the 82586 loads receive buffers in local memory and, after completing the reception, interrupts the Host board to indicate that a packet has been received.

82C501 ETHERNET SERIAL INTERFACE

- — Direct Interface to the 82586 LAN Coprocessor and Ethernet Transceiver
- — Conforms to IEEE 802.3 10BASE5 (Ethernet) and IEEE 802.3 10BASE2 (Cheapernet) Specifications
- — 10 Mb/s Serial Data Rate
- — Manchester Encoding/Decoding and Receive Clock Recovery
- — 10 MHz Transmit Clock Generation
- — Drives and Receives IEEE 802.3 AUI (Transceiver) Cable
- — Optional Watchdog Timer Prevents Babbling
- — Internal Diagnostic Loopback for Fault Detection and Isolation
- — Functionally Compatible with the SEEQ 8023A

The 82C501 provides the Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) Serial Interface for the 82586 LAN Coprocessor. Major functions of the 82C501 include generation of the transmit and receive clock (10 MHz for Ethernet and Cheapernet), Manchester encoding/decoding of transmit and receive data, and interfacing the 10BASE5 Access Unit Interface (AUI/Transceiver) cable. In addition, the 82C501 provides for fault isolation with internal diagnostic loopback. An on-chip watchdog timer prevents the station from locking up in the continuous transmit mode (jabber control).

PC586E Specifications*

Software:	— Network Software Drives are Currently Available for the Following Applications:
	UNIX/TCP-IP
	Novell/Netware
	(Additional Drivers to be Announced)
Hardware:	— IBM PC, PC-XT, PC-AT and Compatible Systems
Cable Connections:	— DB-15 Connector (Ethernet)
	— BNC Connector (Cheapernet)
System Components:	— Intel 82586 LAN Coprocessor
	— Intel 82C501 Ethernet Serial Interface
Memory Capacity:	— Static RAM 16 Kbytes
	— General Address PROM 32 bytes
	— Bootable EPROM 8 Kbytes
Memory Address Ranges:	1. 0C0000h–0C7FFFh 2. 0C8000h–0CFFFFh 3. 0D0000h–0D7FFFh 4. 0D8000h–0DFFFFh 5. F00000h–F3FFFFh 6. F40000h–F7FFFFh 7. F80000h–FBFFFFh 8. FC0000h–FFFFFFh
Frequency:	— Board Master Clock 24 MHz
	— 82586-6 6 MHz
8-Bit PC Bus Frequency (Max.):	— 4.77 MHz 0 Additional Wait-States
	— 8 MHz 0 Additional Wait-States
	— >8 MHz Not Supported
16-Bit AT Bus Frequency (Max.):	— 8 MHz 0 Additional Wait-States
	— 10 MHz 0 Additional Wait-States
	— 12 MHz 1 Additional Wait-States
	— >12 MHz Not Supported
Voltage Limits:	— +5V Input ±5%
	— +12V Input ±5%
Current Requirements:	— +5V Input 3.0A*
	— +12V Input 300 mA*
Power Dissipation:	— Maximum 18.6W*
Temperature Range:	— Operating 0°C to +55°C
	— Storage 0°C to +70°C

DIMENSIONS (Not Including Mounting Bracket)

Length: 8.2 in. (20.8 cm)

Height: 4.2 in. (10.7 cm)

Width: 0.7 in. (1.8 cm)

*Preliminary, subject to change

All information passed between the 82586 and the Host board is made through shared local memory. The Host may load the memory with a command and prompt the 82586 to execute. While receiving a packet, the 82586 loads receive buffers in local memory and, after completing the reception, interrupts the Host board to indicate that a packet has been received.

82C501 ETHERNET SERIAL INTERFACE

- Direct Interface to the 82586 LAN Coprocessor and Ethernet Transceiver
- Conforms to IEEE 802.3 10BASE5 (Ethernet) and IEEE 802.3 10BASE2 (Cheapernet) Specifications
- 10 Mb/s Serial Data Rate
- Manchester Encoding/Decoding and Receive Clock Recovery
- 10 MHz Transmit Clock Generation
- Drives and Receives IEEE 802.3 AUI (Transceiver) Cable
- Optional Watchdog Timer Prevents Babbling
- Internal Diagnostic Loopback for Fault Detection and Isolation
- Functionally Compatible with the SEEQ 8023A

The 82C501 provides the Ethernet (IEEE 802.3 10BASE5) or Cheapernet (IEEE 802.3 10BASE2) Serial Interface for the 82586 LAN Coprocessor. Major functions of the 82C501 include generation of the transmit and receive clock (10 MHz for Ethernet and Cheapernet), Manchester encoding/decoding of transmit and receive data, and interfacing the 10BASE5 Access Unit Interface (AUI/Transceiver) cable. In addition, the 82C501 provides for fault isolation with internal diagnostic loopback. An on-chip watchdog timer prevents the station from locking up in the continuous transmit mode (jabber control).

PC586E Specifications*

Software:	— Network Software Drivers are Currently Available for the Following Applications:	
	UNIX/TCP-IP	
	Novell/Netware	
	(Additional Drivers to be Announced)	
Hardware:	— IBM PC, PC-XT, PC-AT and Compatible Systems	
Cable Connections:	— DB-15 Connector (Ethernet)	
	— BNC Connector (Cheapernet)	
System Components:	— Intel 82586 LAN Coprocessor	
	— Intel 82C501 Ethernet Serial Interface	
Memory Capacity:	— Static RAM	16 Kbytes
	— General Address PROM	32 bytes
	— Bootable EPROM	8 Kbytes
Memory Address Ranges:	1. 0C0000h–0C7FFFh	
	2. 0C8000h–0CFFFFh	
	3. 0D0000h–0D7FFFh	
	4. 0D8000h–0DFFFFh	
	5. F00000h–F3FFFFh	
	6. F40000h–F7FFFFh	
	7. F80000h–FBFFFFh	
	8. FC0000h–FFFFFFh	
Frequency:	— Board Master Clock	24 MHz
	— 82586-6	6 MHz
8-Bit PC Bus Frequency (Max.):	— 4.77 MHz	0 Additional Wait-States
	— 8 MHz	0 Additional Wait-States
	— > 8 MHz	Not Supported
16-Bit AT Bus Frequency (Max.):	— 8 MHz	0 Additional Wait-States
	— 10 MHz	0 Additional Wait-States
	— 12 MHz	1 Additional Wait-States
	— > 12 MHz	Not Supported
Voltage Limits:	— +5V Input ±5%*	
	— +12V Input ±5%*	
Current Requirements:	— +5V Input	3.0A*
	— +12V Input	300 mA*
Power Dissipation:	— Maximum	18.8W*
Temperature Range:	— Operating	0°C to +55°C
	— Storage	0°C to +70°C

DIMENSIONS (Not Including Mounting Bracket)

Length: 8.2 in. (20.8 cm)
Height: 4.2 in. (10.7 cm)
Width: 0.7 in. (1.8 cm)

*Preliminary, subject to change

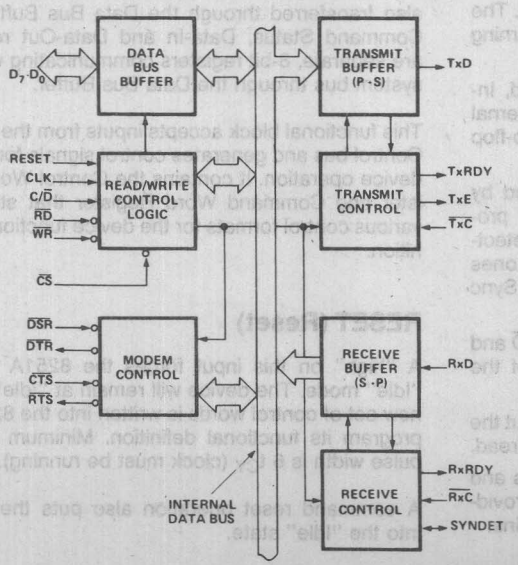
Wide Area Networks

2

2 Wide Area Networks

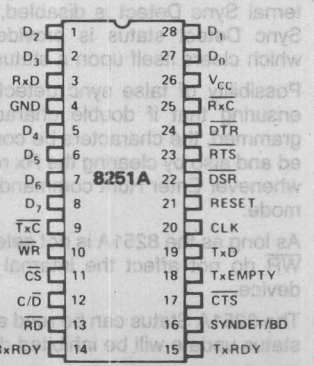
- **1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling**
- **Synchronous Baud Rate—DC to 64K Baud**
- **28-Pin DIP Package**
- **All Inputs and Outputs are TTL Compatible**
- **Available in EXPRESS and Military Versions**

The Intel® 8251A is the industry standard Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's microprocessor families such as MCS-48, 80, 85, and iAPX-86, 88. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is fabricated using Intel's high performance HMOS technology.



205222-1

Figure 1. Block Diagram



205222-2

Figure 2. Pin Configuration

November 1986
Order Number: 205222-002

FEATURES AND ENHANCEMENTS

The 8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the $\overline{RD}$ and $\overline{WR}$ do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.

FUNCTIONAL DESCRIPTION

General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed for a wide range of Intel microcomputers such as 8048, 8080, 8085, 8086 and 8088. Like other I/O devices in a microcomputer system, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support most serial data techniques in use, including IBM "bi-sync".

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

Data Bus Buffer

This 3-state bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The Command Status, Data-In and Data-Out registers are separate, 8-bit registers communicating with the system bus through the Data Bus Buffer.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is 6 t_{CY} (clock must be running).

A command reset operation also puts the device into the "Idle" state.

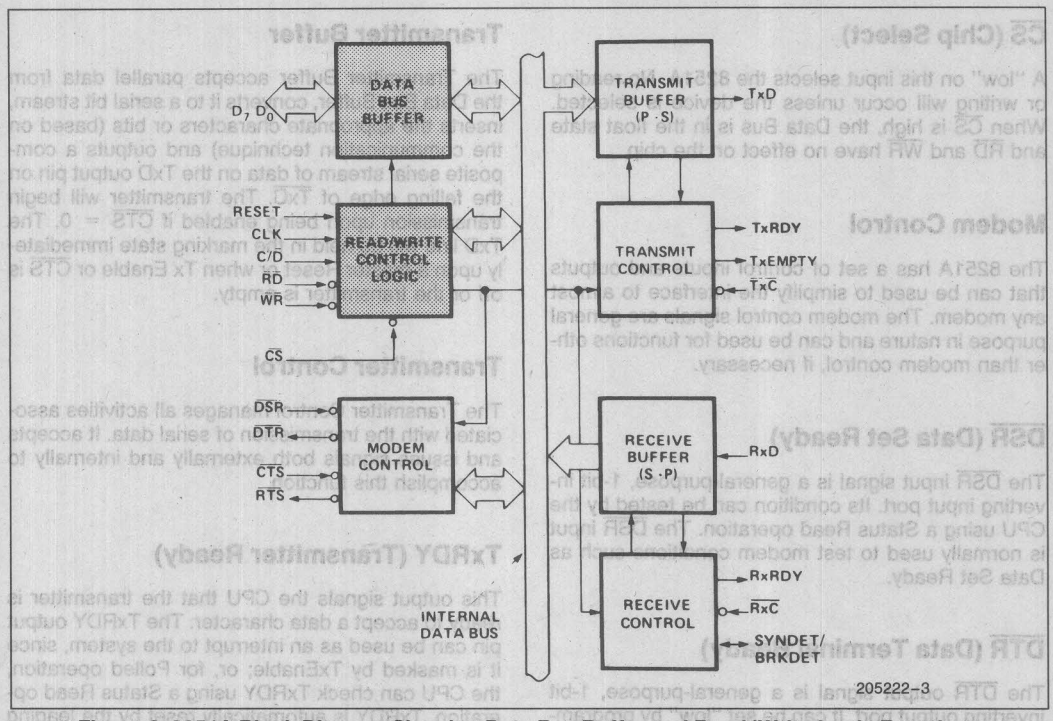


Figure 3. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

CLK (Clock)

The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

$\overline{WR}$ (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

$\overline{RD}$ (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA → DATA BUS
0	1	0	0	DATA BUS → 8251A DATA
1	0	1	0	STATUS → DATA BUS
1	1	0	0	DATA BUS → CONTROL
X	1	1	0	DATA BUS → 3-STATE
X	X	X	1	DATA BUS → 3-STATE

C/$\overline{D}$ (Control/Data)

This input, in conjunction with the $\overline{WR}$ and $\overline{RD}$ inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

1 = CONTROL/STATUS; 0 = DATA.

$\overline{CS}$ (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When $\overline{CS}$ is high, the Data Bus is in the float state and $\overline{RD}$ and $\overline{WR}$ have no effect on the chip.

Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

$\overline{DSR}$ (Data Set Ready)

The $\overline{DSR}$ input signal is a general-purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The $\overline{DSR}$ input is normally used to test modem conditions such as Data Set Ready.

$\overline{DTR}$ (Data Terminal Ready)

The $\overline{DTR}$ output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{DTR}$ output signal is normally used for modem control such as Data Terminal Ready.

$\overline{RTS}$ (Request to Send)

The $\overline{RTS}$ output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The $\overline{RTS}$ output signal is normally used for modem control such as Request to Send.

$\overline{CTS}$ (Clear to Send)

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one". If either a Tx Enable off or $\overline{CTS}$ off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.

Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of $\overline{TxC}$. The transmitter will begin transmission upon being enabled if $\overline{CTS} = 0$. The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable or $\overline{CTS}$ is off or the transmitter is empty.

Transmitter Control

The Transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

TxRDY (Transmitter Ready)

This output signals the CPU that the transmitter is ready to accept a data character. The TxRDY output pin can be used as an interrupt to the system, since it is masked by TxEnable; or, for Polled operation, the CPU can check TxRDY using a Status Read operation. TxRDY is automatically reset by the leading edge of $\overline{WR}$ when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxRDY status bit is *not* masked by TxEnable, but will only indicate the Empty/Full Status of the Tx Data Input Register.

TxE (Transmitter Empty)

When the 8251A has no characters to send, the TxEMPTY output will go "high". It resets upon receiving a character from CPU if the transmitter is enabled. TxEMPTY remains high when the transmitter is disabled. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplex operational mode.

In the Synchronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers". Tx EMPTY does not go low when the SYNC characters are being shifted out.

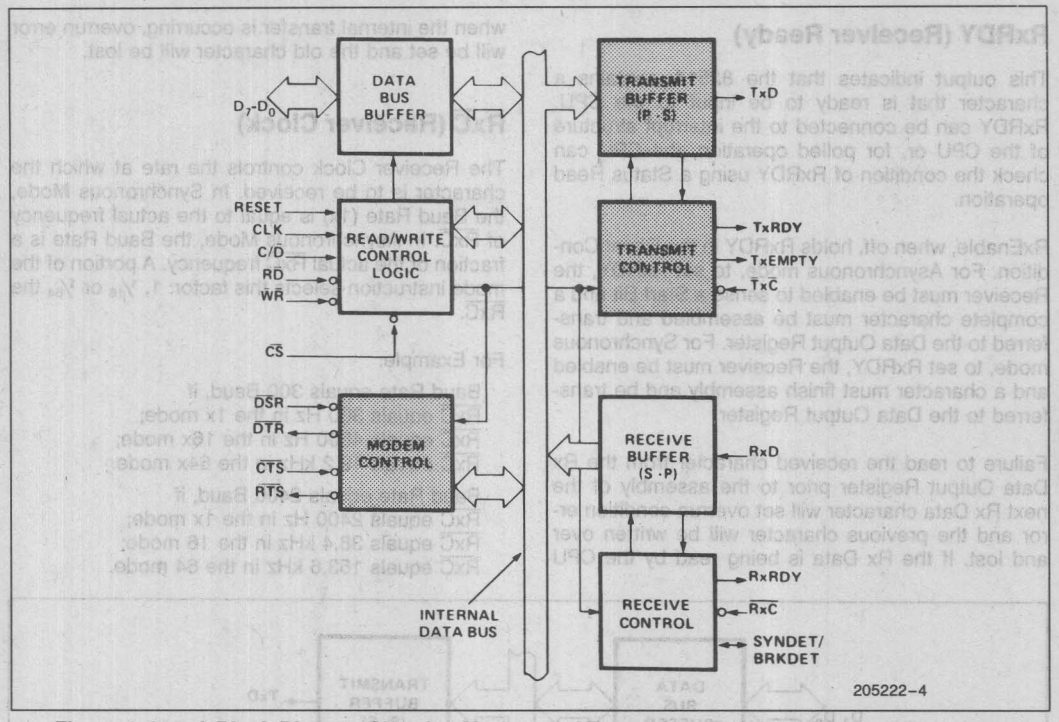


Figure 4. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions

TxC (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the TxC frequency. In Asynchronous transmission mode, the baud rate is a fraction of the actual TxC frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the TxC.

For Example:

If Baud Rate equals 110 Baud,
TxC equals 110 Hz in the 1x mode.
TxC equals 1.72 kHz in the 16x mode.
TxC equals 7.04 kHz in the 64x mode.

The falling edge of TxC shifts the serial data out of the 8251A.

Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of RxC.

Receiver Control

This functional block manages all receiver-related activities which consists of the following features.

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition". Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the normal center of the Start bit (RxD = low).

Parity error detection sets the corresponding status bit.

The Framing Error status bit is set if the Stop bit is absent at the end of the data byte (asynchronous mode).

RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. RxRDY can be connected to the interrupt structure of the CPU or, for polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

RxEnable, when off, holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

$\overline{\text{RxC}}$ (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of $\overline{\text{RxC}}$. In Asynchronous Mode, the Baud Rate is a fraction of the actual $\overline{\text{RxC}}$ frequency. A portion of the mode instruction selects this factor: 1, 1/16 or 1/64 the $\overline{\text{RxC}}$.

For Example:

Baud Rate equals 300 Baud, if
$\overline{\text{RxC}}$ equals 300 Hz in the 1x mode;
$\overline{\text{RxC}}$ equals 4800 Hz in the 16x mode;
$\overline{\text{RxC}}$ equals 19.2 kHz in the 64x mode.

Baud Rate equals 2400 Baud, if
$\overline{\text{RxC}}$ equals 2400 Hz in the 1x mode;
$\overline{\text{RxC}}$ equals 38.4 kHz in the 16 mode;
$\overline{\text{RxC}}$ equals 153.6 kHz in the 64 mode.

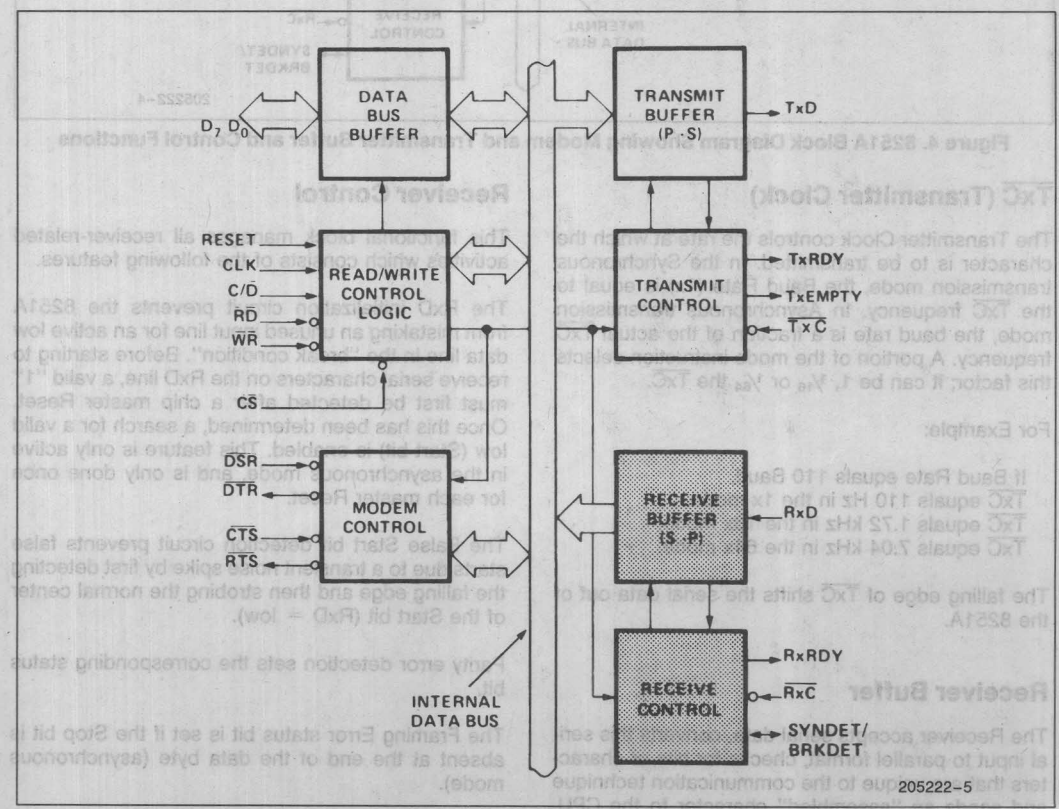


Figure 5. 8251A Block Diagram Showing Receiver Buffer and Control Functions

Data is sampled into the 8251A on the rising edge of $\overline{RxC}$.

NOTE:
In most communication systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both $\overline{TxC}$ and $\overline{RxC}$ will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

SYNDET (SYNC Detect/ BRKDET Break Detect)

This pin is used in Synchronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bi-sync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next $\overline{RxC}$. Once in SYNC, the "high" input signal can be removed. When External SYNC Detect is programmed, Internal SYNC Detect is disabled.

BREAK (Async Mode Only)

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including the start bits, data bits, and parity bits). Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

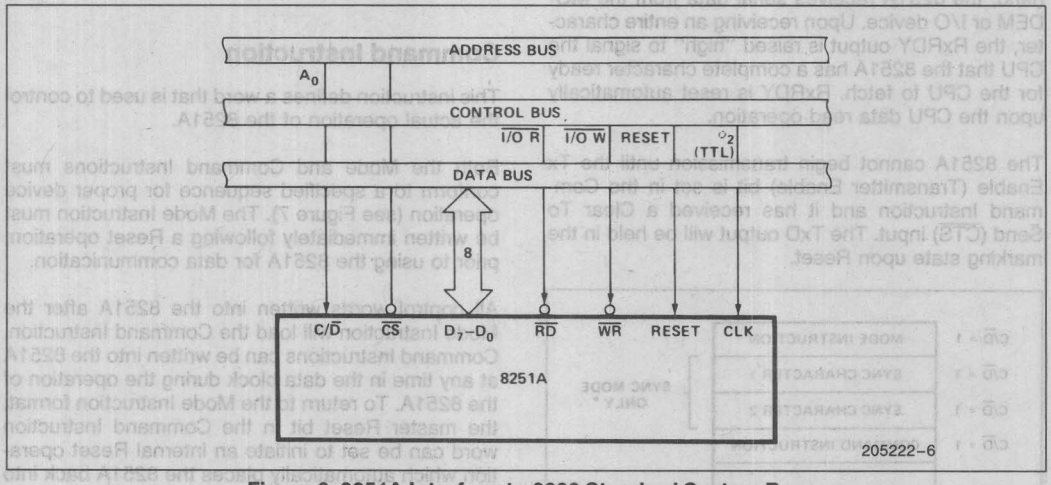


Figure 6. 8251A Interface to 8080 Standard System Bus

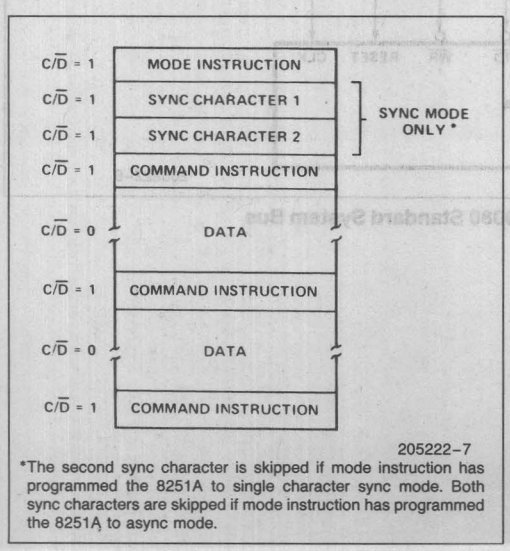
DETAILED OPERATION DESCRIPTION

General

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send ($\overline{CTS}$) input. The TxD output will be held in the marking state upon Reset.



*The second sync character is skipped if mode instruction has programmed the 8251A to single character sync mode. Both sync characters are skipped if mode instruction has programmed the 8251A to async mode.

Figure 7. Typical Data Block

Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

Mode Instruction

This instruction defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be written.

Command Instruction

This instruction defines a word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation (see Figure 7). The Mode Instruction must be written immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instruction or Sync characters.

Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components, one Asynchronous and the other Synchronous, sharing the same package. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

NOTE:
When parity is enabled it is not considered as one of the data bits for the purpose of programming word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of $\overline{TxC}$ at a rate equal to 1, 1/16, or 1/64 that of the $\overline{TxC}$, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains "high" (marking) unless a Break (continuously low) has been programmed.

Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of the $\overline{RxC}$. If a low level is detected as the STOP bit the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the *receiver* requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxRDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the OVERRUN Error flag

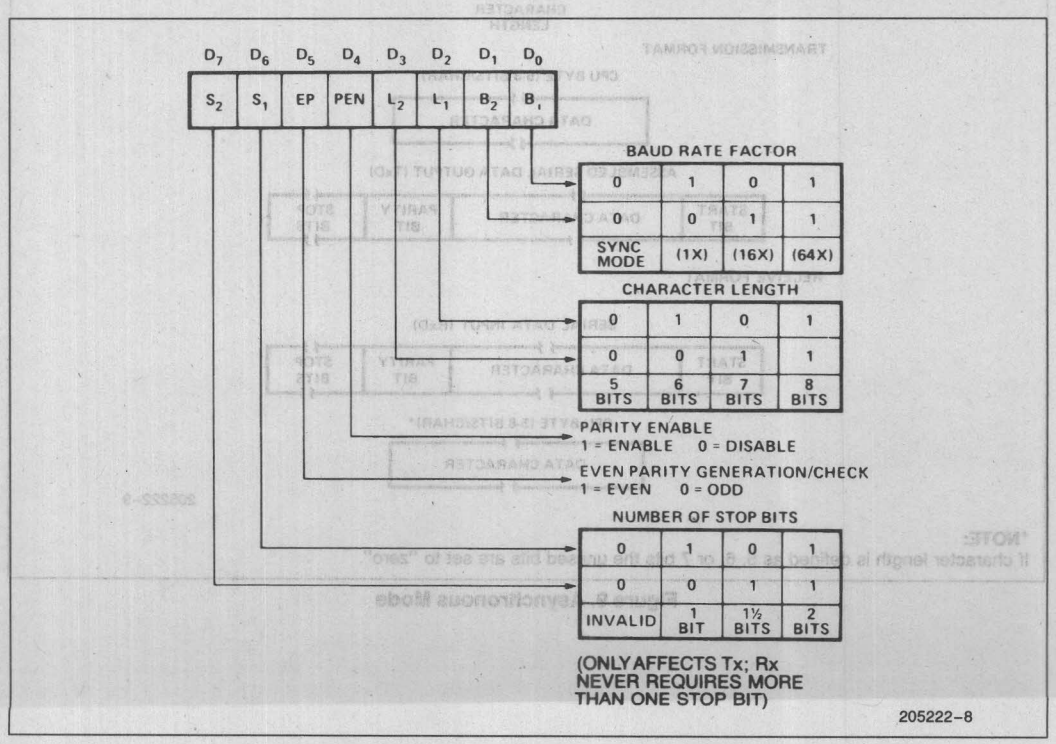


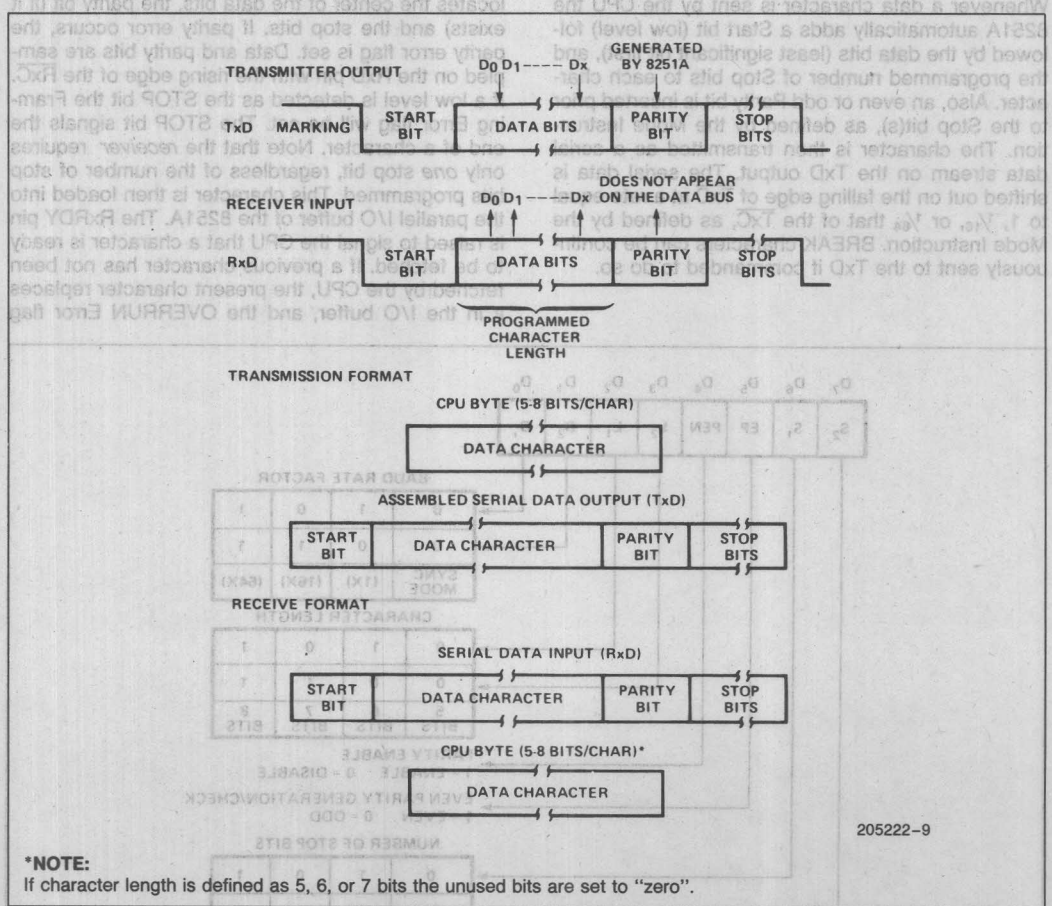
Figure 8. Mode Instruction Format, Asynchronous Mode

is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

Synchronous Mode (Transmission)

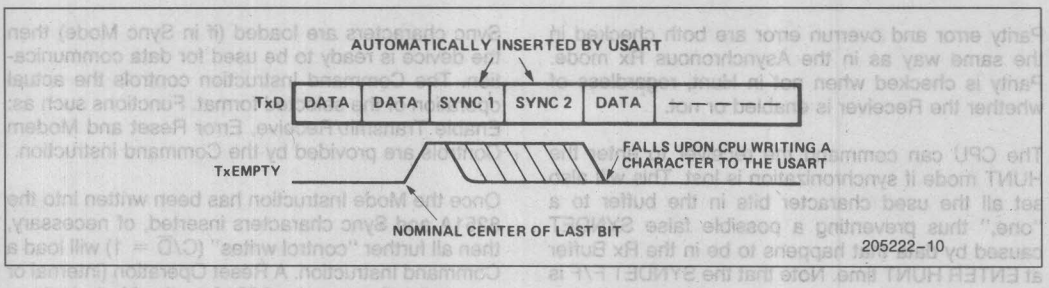
The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the $\overline{CTS}$ line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of $\overline{TxC}$. Data is shifted out at the same rate as the $\overline{TxC}$.

Once transmission has started, the data stream at the TxD output must continue at the $\overline{TxC}$ rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



*NOTE:
If character length is defined as 5, 6, or 7 bits the unused bits are set to "zero".

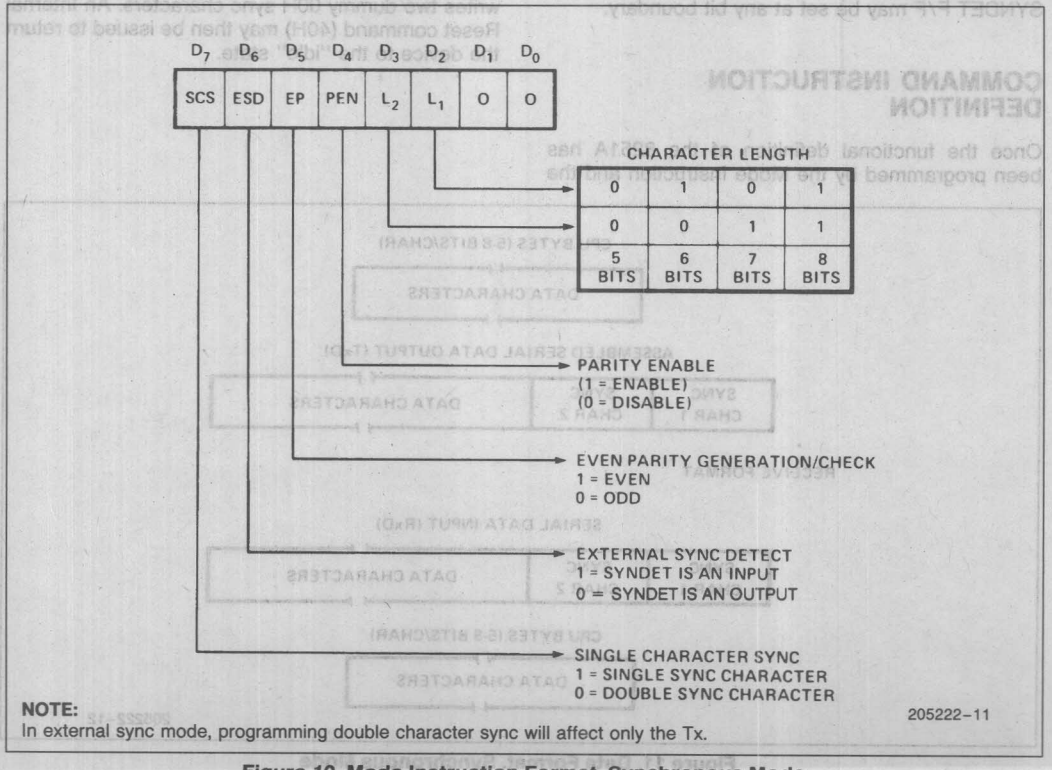
Figure 9. Asynchronous Mode



Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled on the rising edge of $\overline{RxC}$. The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one $\overline{RxC}$ cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.



NOTE:
In external sync mode, programming double character sync will affect only the Tx.

Figure 10. Mode Instruction Format, Synchronous Mode

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one," thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication). When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the Sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, of necessary, then all further "control writes" ($C/\overline{D} = 1$) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

NOTE:
Internal Reset on Power-up:

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with $C/\overline{D} = 1$ configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "idle" state.

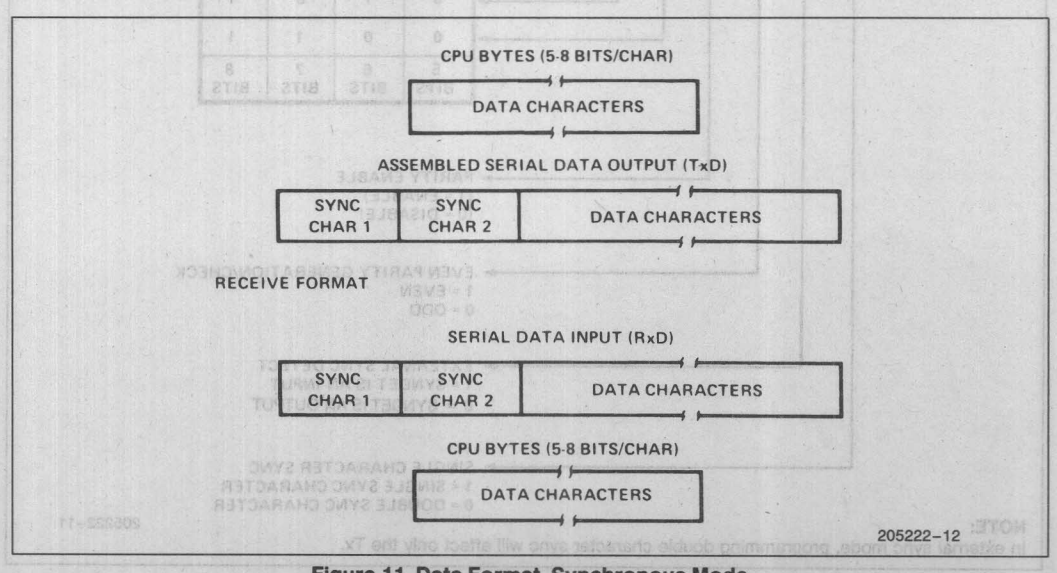
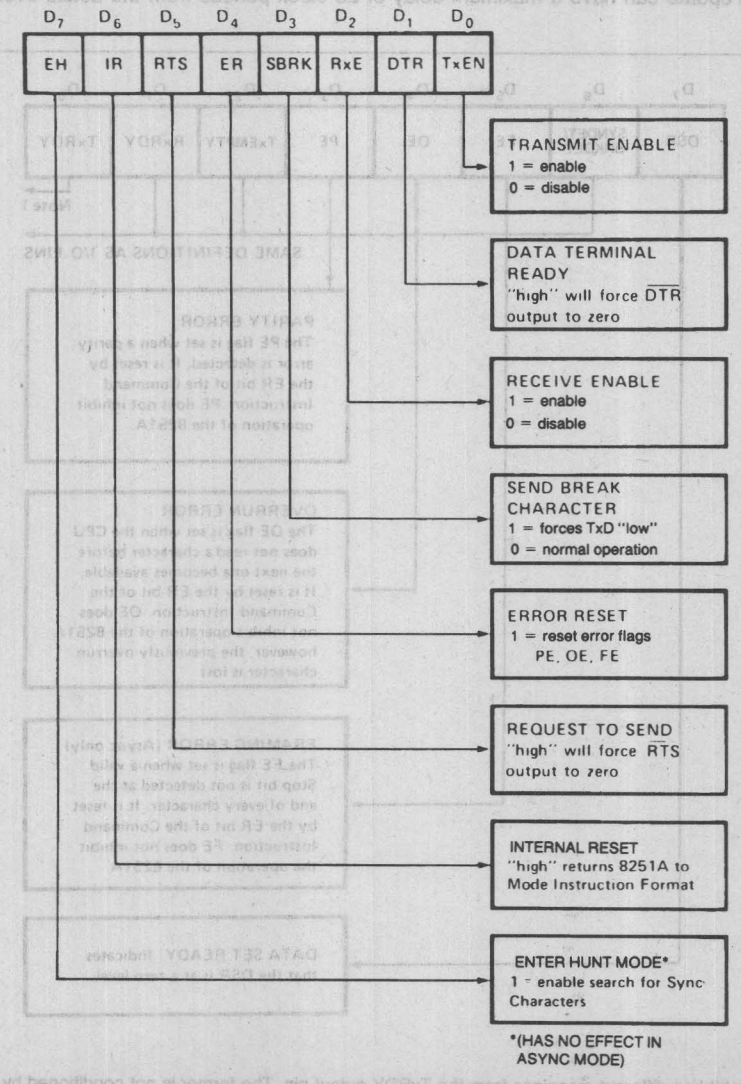


Figure 11. Data Format, Synchronous Mode



NOTE:
Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 12. Command Instruction Format

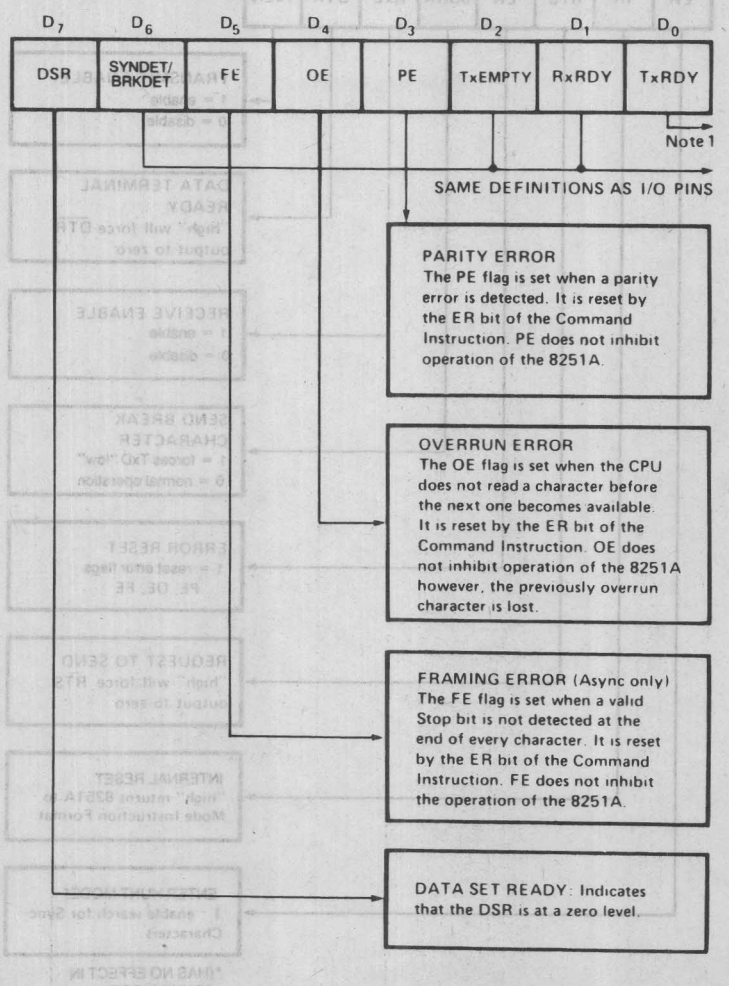
STATUS READ DEFINITION

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (Status update is inhibited during status read.)

A normal "read" command is issued by the CPU with C/$\overline{D}$ = 1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely polled or interrupt-driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.



NOTE:
1. TxRDY status bit has different meanings from the TxRDY output pin. The former is not conditioned by $\overline{CTS}$ and TxEN; the latter is conditioned by both $\overline{CTS}$ and TxEN.
i.e. TxRDY status bit = DB Buffer Empty
TxRDY pin out = DB Buffer Empty • ($\overline{CTS}$ = 0) • (TxEN = 1)

Figure 13. Status Read Format

APPLICATIONS OF THE 8251A

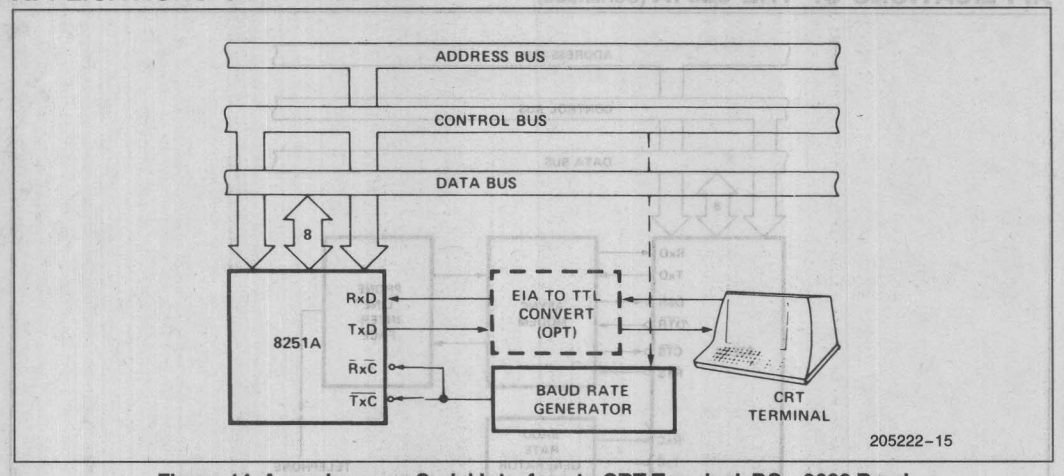


Figure 14. Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud

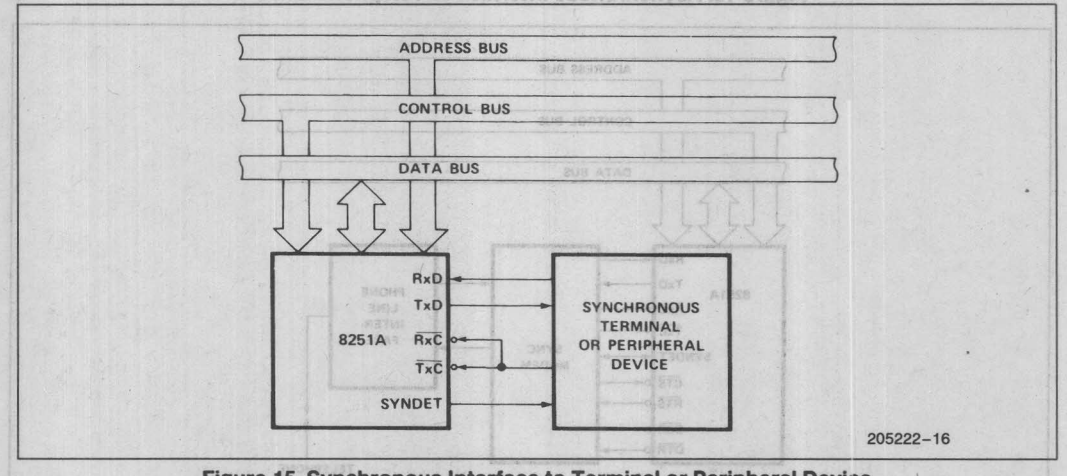


Figure 15. Synchronous Interface to Terminal or Peripheral Device

APPLICATIONS OF THE 8251A (Continued)

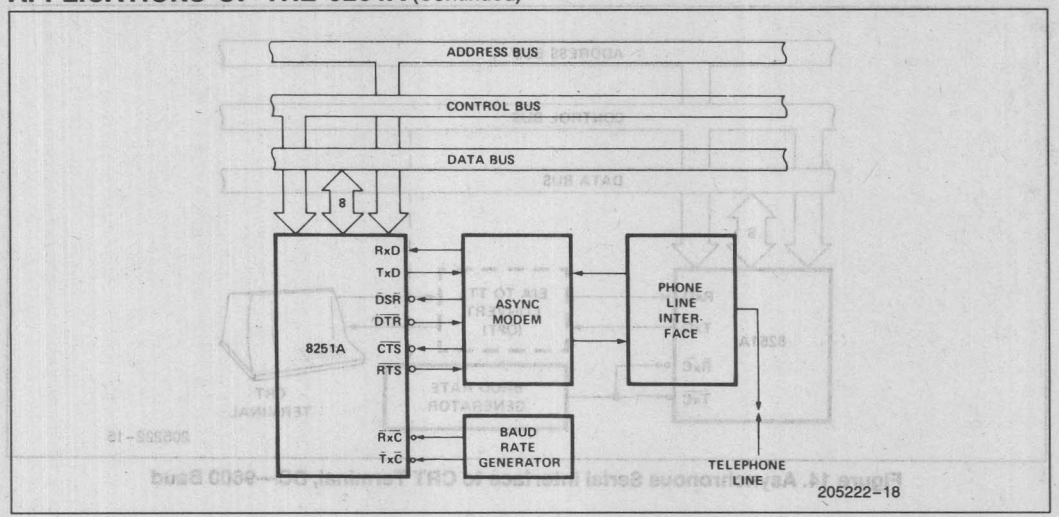


Figure 16. Asynchronous Interface to Telephone Lines

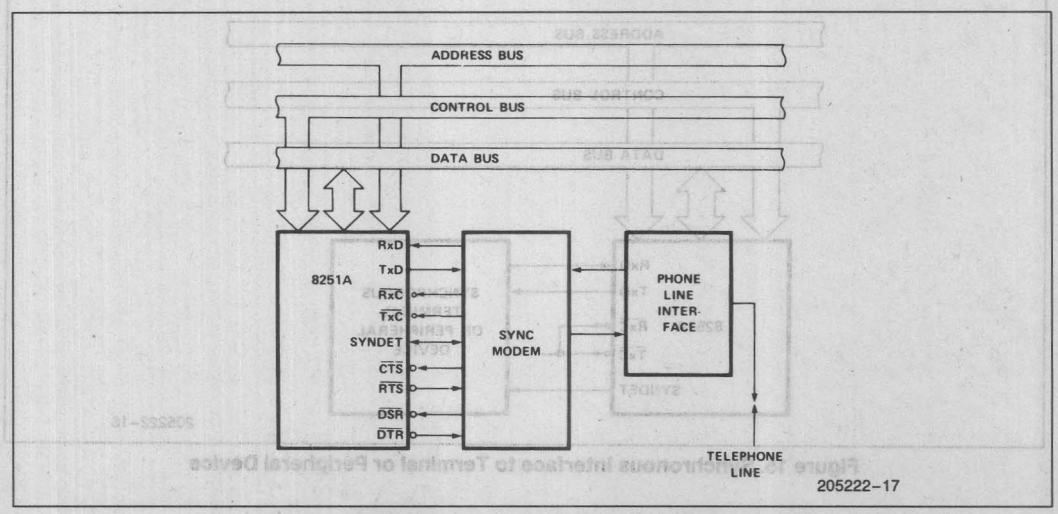


Figure 17. Synchronous Interface to Telephone Lines

NOTES:
1. AC timings measured V_{OH} = 2.0 V_{OL} = 0.8, and with load circuit of Figure 18.
2. Chip Select (CS) and Command/Data (C/D) are considered as Addresses.
3. Assumes that Address is valid before $R_D \downarrow$.
4. This recovery time is for Mode Initialization only. Write Data is allowed only when TxRDY = 1. Recovery Time between Writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 16 t_{CY}.
5. The TxC and RxC frequencies have the following limitations with respect to CLK: For 1x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(30\ t_{CY})$: For 16x and 64x Baud Rate, f_{Tx} or $f_{Rx} \leq 1/(4.5\ t_{CY})$. This applies to Baud Rates less than or equal to 64K Baud.
6. Reset Pulse Width = 6 t_{CY} minimum; System clock must be running during Reset.
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.
8. In external sync mode the tes spec. requires the ratio of the system clock (clock) to receive or transmit bit ratios to be *greater than 34*.
9. A float is defined as the point where the data bus falls below a logic 1 (2.0V @ I_{OH} limit) or rises above a Logic 0 (0.8V @ I_{OL} limit).

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin
with Respect to Ground.....−0.5V to +7V
Power Dissipation 1W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V ±10%, GND = 0V*

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
I_{OFL}	Output Float Leakage		±10	μA	V_{OUT} = V_{CC} to 0.45V
I_{IL}	Input Leakage		±10	μA	V_{IN} = V_{CC} to 0.45V
I_{CC}	Power Supply Current		100	ma	All Outputs = High

CAPACITANCE T_A = 25°C, V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Unit	Test Conditions
C_{IN}	Input Capacitance		10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V ±10%, GND = 0V*

Bus Parameters (Note 1)

READ CYCLE

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AR}	Address Stable Before $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	(Note 2)
t_{RA}	Address Hold Time for $\overline{READ}$ ($\overline{CS}$, C/$\overline{D}$)	0		ns	(Note 2)
t_{RR}	$\overline{READ}$ Pulse Width	250		ns	
t_{RD}	Data Delay from $\overline{READ}$		200	ns	3, C_L = 150 pF
t_{DF}	$\overline{READ}$ to Data Floating	10	100	ns	(Note 1, 9)

WRITE CYCLE

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	Address Stable Before $\overline{WRITE}$	0		ns	
t_{WA}	Address Hold Time for $\overline{WRITE}$	0		ns	
t_{WW}	$\overline{WRITE}$ Pulse Width	250		ns	
t_{DW}	Data Set-Up Time for $\overline{WRITE}$	150		ns	
t_{WD}	Data Hold Time for $\overline{WRITE}$	20		ns	
t_{RV}	Recovery Time Between WRITES	6		t_{CY}	(Note 4)

A.C. CHARACTERISTICS (Continued)

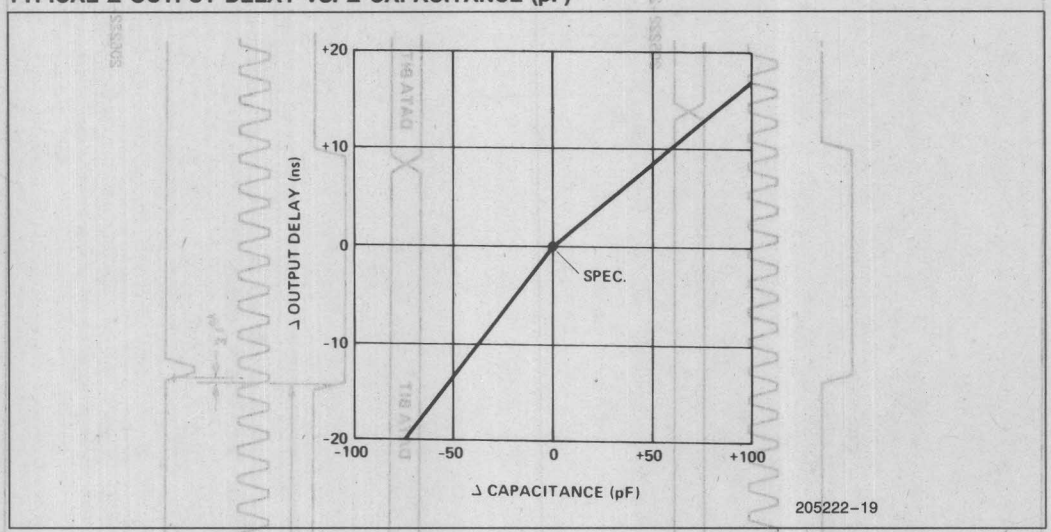
OTHER TIMINGS

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{CY}	Clock Period	320	1350	ns	(Note 5, 6)
t_{ϕ}	Clock High Pulse Width	120	$t_{CY}-90$	ns	
$\overline{t_{\phi}}$	Clock Low Pulse Width	90		ns	
t_R, t_F	Clock Rise and Fall Time		20	ns	
t_{DTx}	TxD Delay from Falling Edge of $\overline{TxC}$		1	μs	
f_{Tx}	Transmitter Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{TPW}	Transmitter Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{TPD}	Transmitter Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
f_{Rx}	Receiver Input Clock Frequency				
	1x Baud Rate	DC	64	kHz	
	16x Baud Rate	DC	310	kHz	
	64x Baud Rate	DC	615	kHz	
t_{RPW}	Receiver Input Clock Pulse Width				
	1x Baud Rate	12		t_{CY}	
	16x and 64x Baud Rate	1		t_{CY}	
t_{RPD}	Receiver Input Clock Pulse Delay				
	1x Baud Rate	15		t_{CY}	
	16x and 64x Baud Rate	3		t_{CY}	
t_{TxRDY}	TxRDY Pin Delay from Center of Last Bit		14	t_{CY}	(Note 7)
$t_{TxRDY\ CLEAR}$	TxRDY $\downarrow$ from Leading Edge of $\overline{WR}$		400	ns	(Note 7)
t_{RxRDY}	RxRDY Pin Delay from Center of Last Bit		26	t_{CY}	(Note 7)
$t_{RxRDY\ CLEAR}$	RxRDY $\downarrow$ from Leading Edge of $\overline{RD}$		400	ns	(Note 7)
t_{IS}	Internal SYNDET Delay from Rising Edge of $\overline{RxC}$		26	t_{CY}	(Note 7)
t_{ES}	External SYNDET Set-Up Time After Rising Edge of $\overline{RxC}$	16 t_{CY}	$t_{RPD}-t_{CY}$	ns	(Note 7)
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit		20	t_{CY}	(Note 7)
t_{WC}	Control Delay from Rising Edge of WRITE (TxEn, $\overline{DTR}$, $\overline{RTS}$)		8	t_{CY}	(Note 7)
t_{CR}	Control to READ Set-Up Time ($\overline{DSR}$, $\overline{CTS}$)	20		t_{CY}	(Note 7)

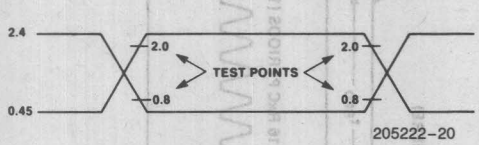
***NOTE:**
For Extended Temperature EXPRESS, use MIL 8251A electrical parameters.

A.C. CHARACTERISTICS (Continued)

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE (pF)



A.C. TESTING INPUT, OUTPUT WAVEFORM



AC Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

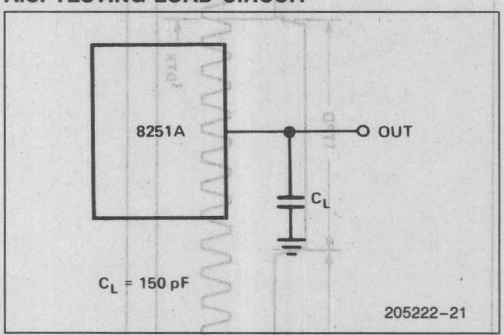
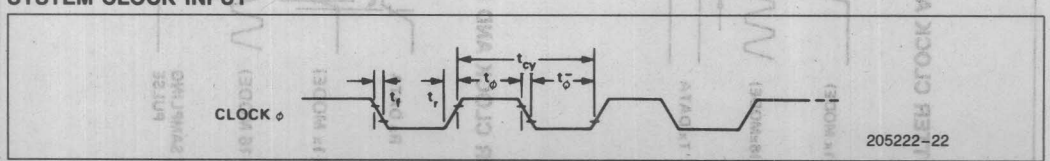


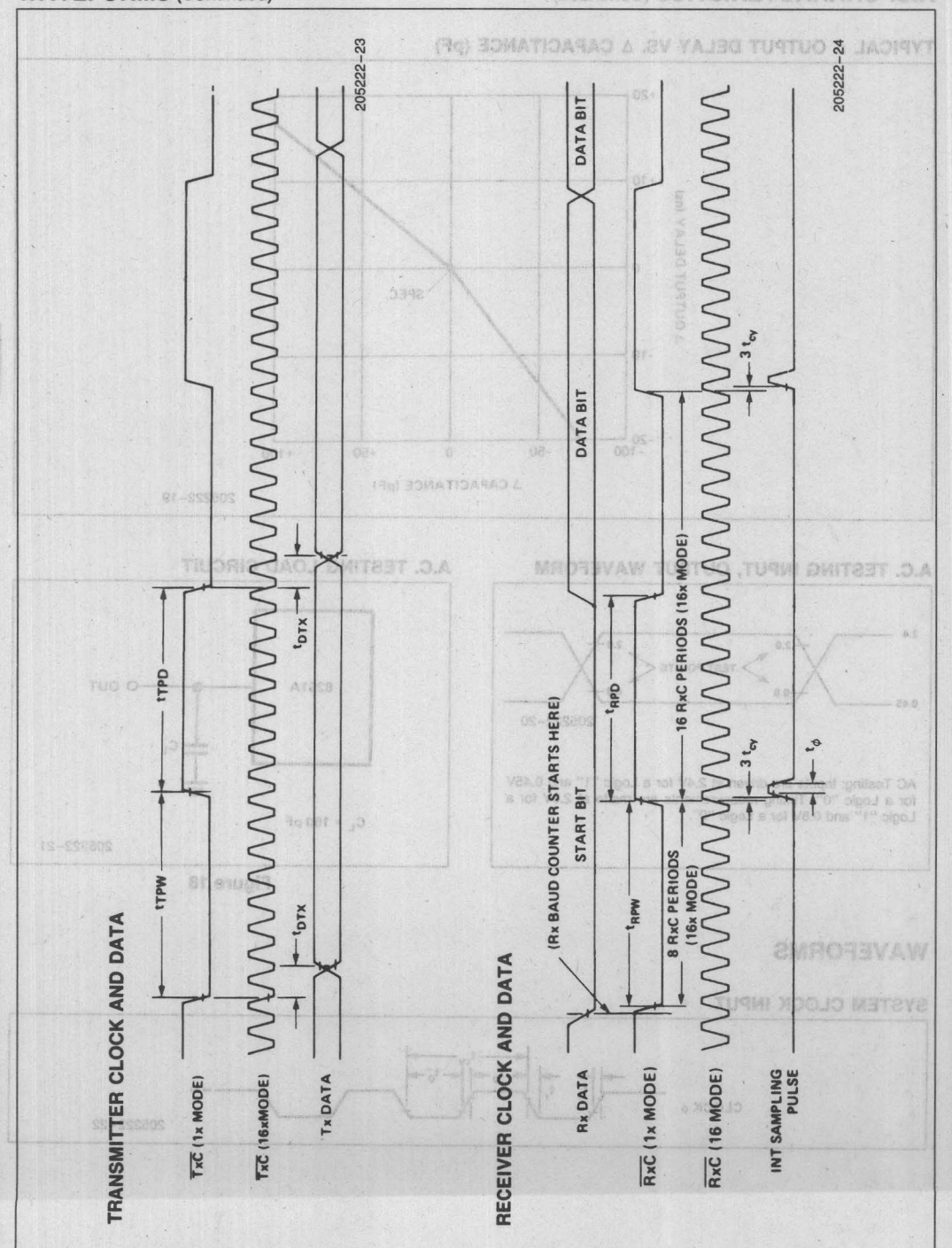
Figure 18

WAVEFORMS

SYSTEM CLOCK INPUT

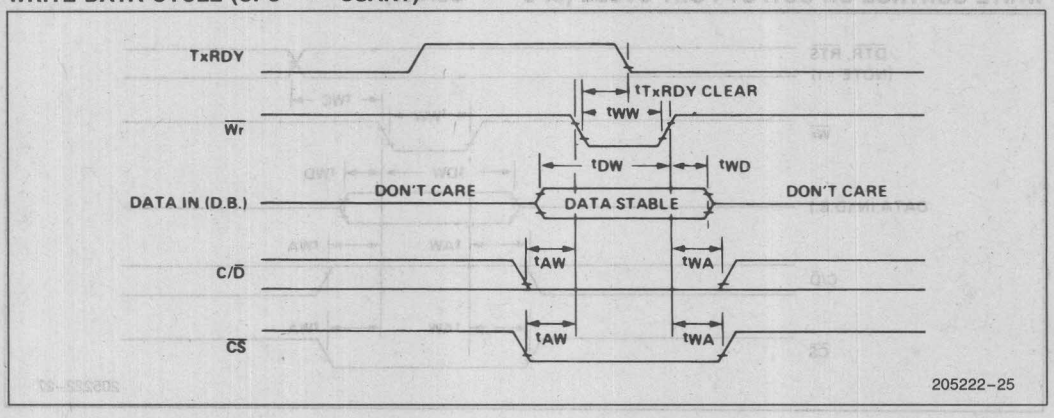


WAVEFORMS (Continued)

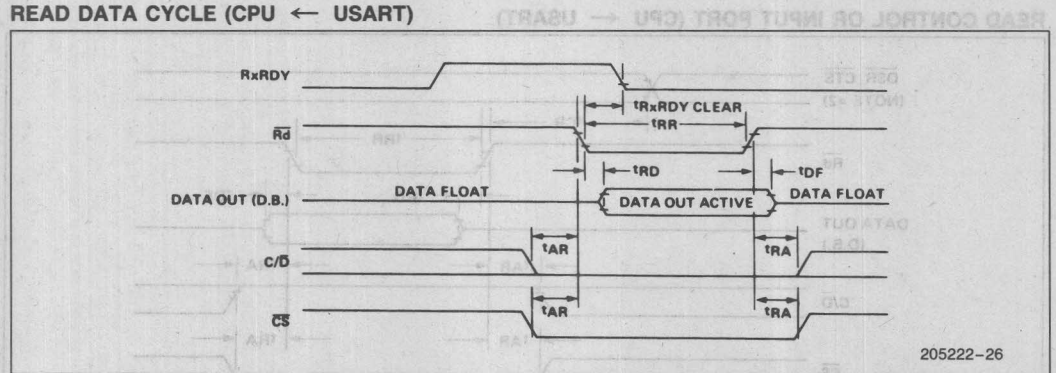


WAVEFORMS (Continued)

WRITE DATA CYCLE (CPU → USART)



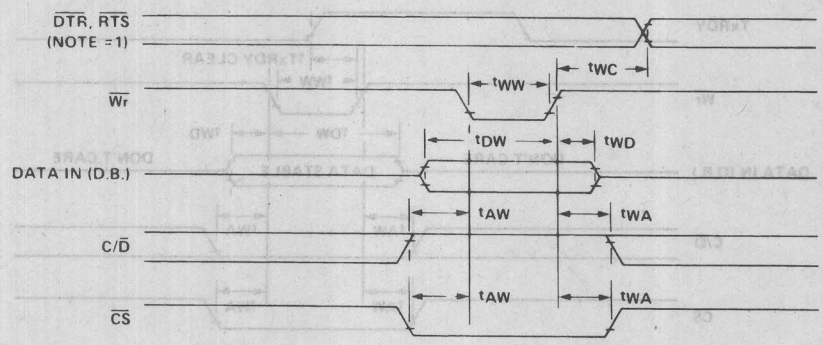
READ DATA CYCLE (CPU ← USART)



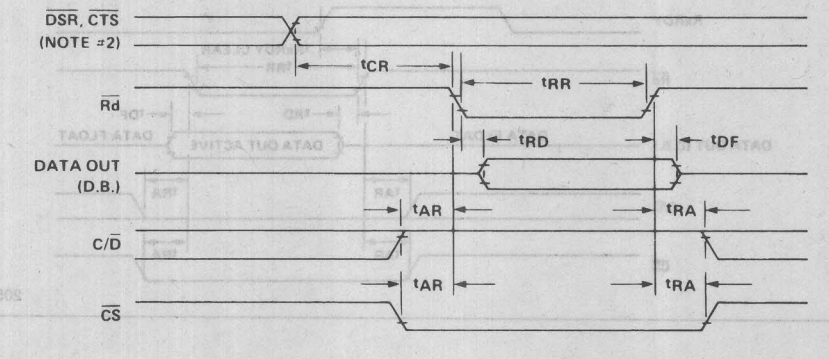
2

WAVEFORMS (Continued)

WRITE CONTROL OR OUTPUT PORT CYCLE (CPU → USART)

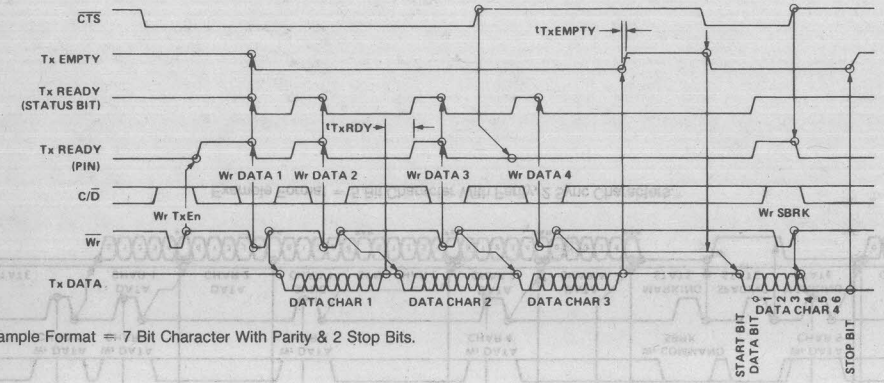


READ CONTROL OR INPUT PORT (CPU ← USART)



NOTES:
1. T_{WC} includes the response timing of a control byte.
2. T_{CR} includes the effect of CTS on the TxENBL circuitry.

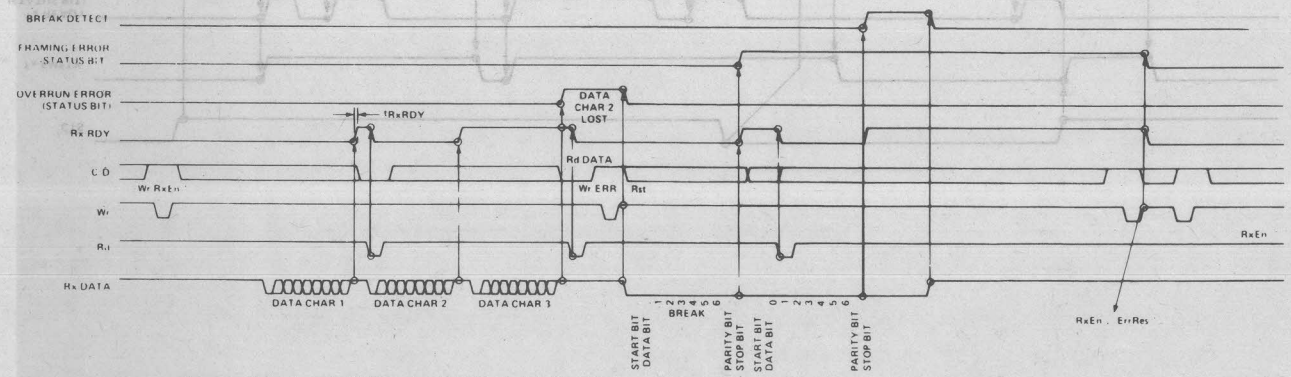
TRANSMITTER CONTROL AND FLAG TIMING (ASYNC MODE)



Example Format = 7 Bit Character With Parity & 2 Stop Bits.

205222-29

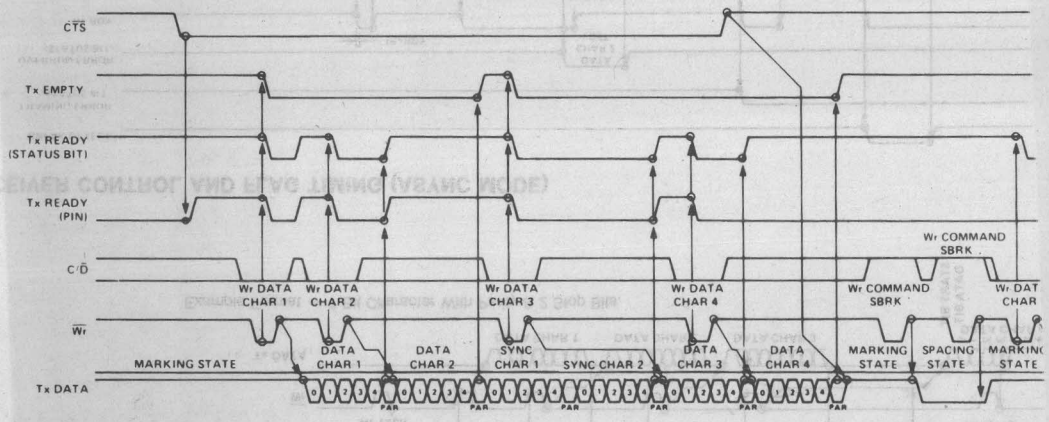
RECEIVER CONTROL AND FLAG TIMING (ASYNC MODE)



Example Format = 7 Bit Character With Parity & 2 Stop Bits

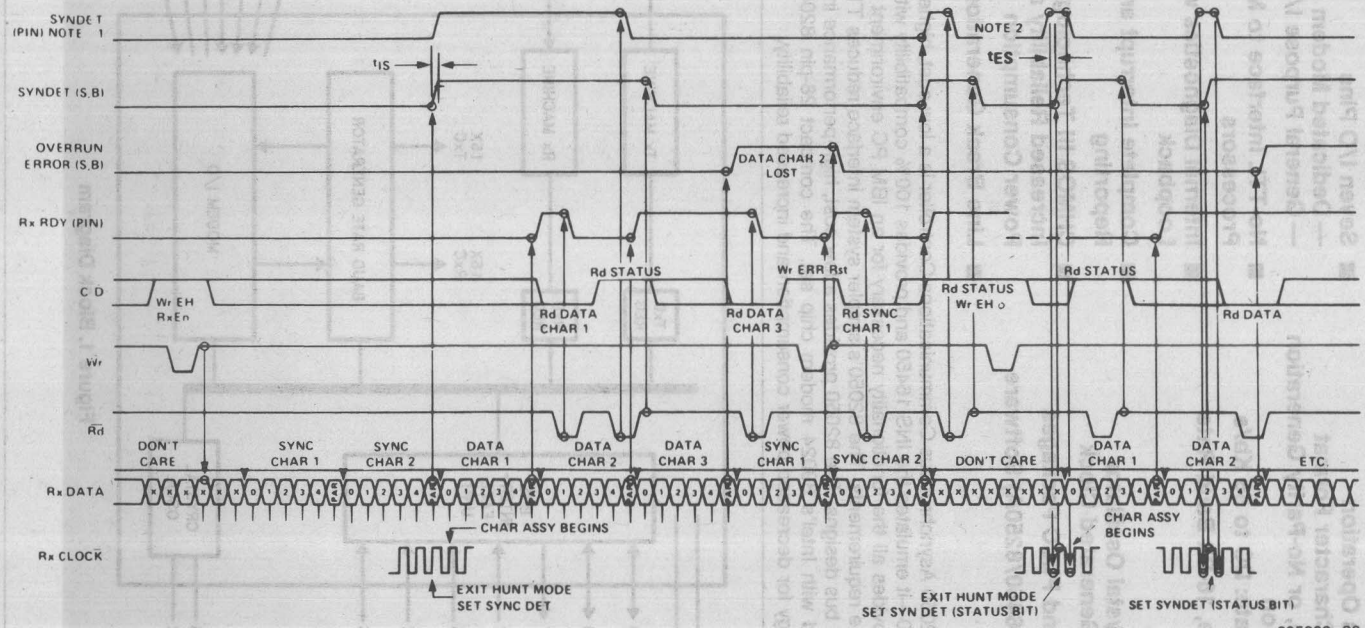
205222-30

TRANSMITTER CONTROL AND FLAG TIMING (SYNC MODE)



Example Format = 5 Bit Character With Parity, 2 Sync Characters.

RECEIVER CONTROL AND FLAG TIMING (SYNC MODE)



205222-32

NOTES:
1. Internal Sync, 2 Sync Characters, 5 Bits With Parity.
2. External Sync, 5 Bits, With Parity.

82050
ASYNCHRONOUS COMMUNICATIONS CONTROLLER

- **Asynchronous Operation**
  - **— 5- to 8-Bit Character Format**
  - **— Odd-, Even-, or No-Parity Generation and Detection**
  - **— Serial Bit Rate: DC to 56 Kb/s**
- **Programmable, 16-Bit Baud Rate Generator**
- **System Clock**
  - **— On-Chip Crystal Oscillator**
  - **— Externally Generated Clock**
- **28-Lead DIP and PLCC Packages**
- **IBM PC (INS 16450/8250A) Software Compatible**
- **Seven I/O Pins**
  - **— Dedicated Modem I/O**
  - **— General Purpose I/O**
- **No-TTL Interface to Most Intel Processors**
- **Internal Diagnostics with Local Loopback**
- **Complete Interrupt and Status Reporting**
- **CHMOS III Technology Provides Increased Reliability and Reduced Power Consumption**
- **Line Break Generation and Detection**

The Intel CHMOS 82050 Asynchronous Communications Controller is a low cost, higher performance alternative to the INS 16450—it emulates the INS 16450 and provides 100% compatibility with IBM PC software. Its 28-lead package provides all the functionality necessary for an IBM PC environment while substantially decreasing board space requirements. The 82050's simpler system interface reduces TTL glue—especially for higher frequency PC bus designs. The 82050 provides a low cost, high-performance integrated modem solution when combined with Intel's 89024 modem chip set. The compact 28-pin 82050 is fabricated using CHMOS III technology for decreased power consumption and increased reliability.

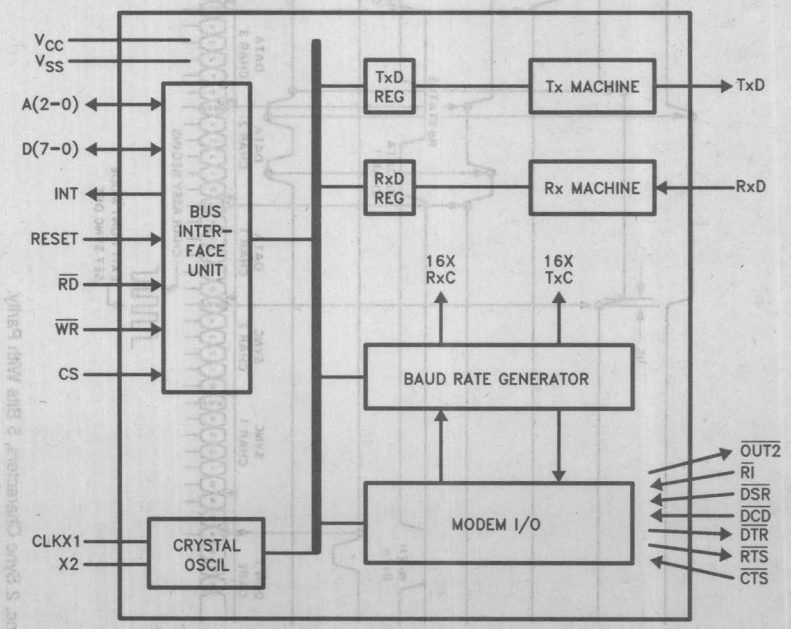


Figure 1. Block Diagram

August 1990
Order Number: 290137-004

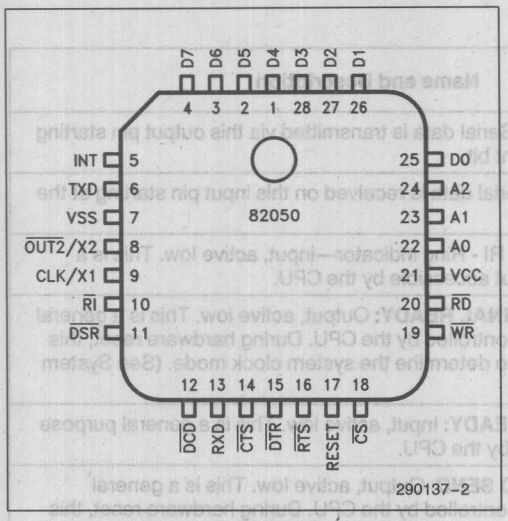


Figure 2. PLCC Pinout

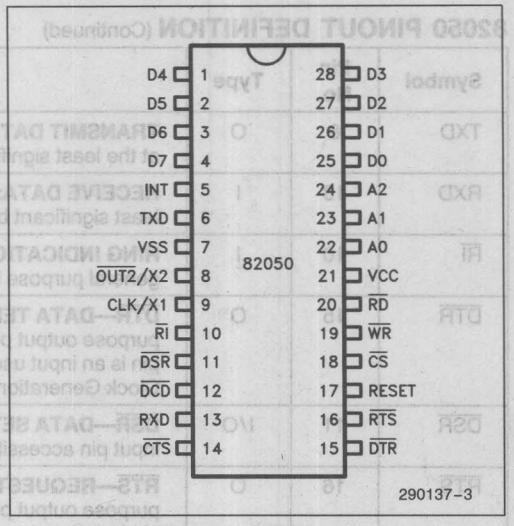


Figure 3. DIP Pinout

82050 PINOUT DEFINITION

Symbol	Pin No.	Type	Name and Description
RESET	17	I	**RESET:** A high on this input pin resets the 82050.
$\overline{CS}$	18	I	**CHIP SELECT:** A low on this input pin enables the 82050 and allows read or write operations.
A2–A0	24–22	I	**ADDRESS PINS:** These inputs interface with three bits of the system address bus to select one of the internal registers for read or write.
D7–D0	1–4 25–28	I/O	**DATA BUS:** Bi-directional, three state, 8-Bit Data Bus. These pins allow transfer of bytes between the CPU and the 82050.
$\overline{RD}$	20	I	**READ:** A low on this input pin allows the CPU to read data or status bytes from the 82050.
$\overline{WR}$	19	I	**WRITE:** A low on this input allows the CPU to write data or control bytes to the 82050.
INT	5	O	**INTERRUPT:** A high on this output pin signals an interrupt request to the CPU. The CPU may determine the particular source and cause of the interrupt by reading the 82050 status registers.
CLK/X1	9	I	**MULTIFUNCTION:** This input pin serves as a source for the internal system clock. The clock may be asynchronous to the serial clocks and to the processor clock. This pin may be used in one of two modes: CLK-in this mode an externally generated clock should be used to drive this input pin; X1-in this mode the clock is generated by a crystal to be connected between this pin (X1) and the X2 pin. (See system clock generation.)
$\overline{OUT2}$/X2	8	O	**MULTIFUNCTION:** This is a dual-function pin which may be configured to one of the following functions: $\overline{OUT2}$—a general purpose output pin controlled by the CPU is only available when the CLK/X1 pin is driven by an externally generated clock; X2—this pin serves as an output pin for the crystal oscillator. Note: The configuration of pin is done during hardware reset. For more details refer to the system clock generation.

82050 PINOUT DEFINITION (Continued)

Symbol	Pin No.	Type	Name and Description
TXD	6	O	**TRANSMIT DATA:** Serial data is transmitted via this output pin starting at the least significant bit.
RXD	13	I	**RECEIVE DATA:** Serial data is received on this input pin starting at the least significant bit.
$\overline{RI}$	10	I	**RING INDICATION:** RI - Ring indicator—input, active low. This is a general purpose input accessible by the CPU.
$\overline{DTR}$	15	O	**$\overline{DTR}$—DATA TERMINAL READY:** Output, active low. This is a general purpose output pin controlled by the CPU. During hardware reset, this pin is an input used to determine the system clock mode. (See System Clock Generation.)
$\overline{DSR}$	11	I/O	**$\overline{DSR}$—DATA SET READY:** Input, active low. This is a general purpose input pin accessible by the CPU.
$\overline{RTS}$	16	O	**$\overline{RTS}$—REQUEST TO SEND:** Output, active low. This is a general purpose output pin controlled by the CPU. During hardware reset, this pin is an input used to determine the system clock mode. (See system clock generation)
$\overline{CTS}$	14	I	**CLEAR TO SEND:** Input active low. This is a general purpose input pin accessible by the CPU.
$\overline{DCD}$	12	I/O	**$\overline{DCD}$—DATA CARRIER DETECTED:** Input, active low. This is a general purpose input pin accessible by the CPU.
VCC	21	P	**VCC:** Device power supply.
VSS	7	P	**VSS:** Ground.

SYSTEM INTERFACE

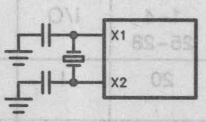
The 82050 has a simple demultiplexed bus interface which consists of a bidirectional, three-state, 8-bit data bus and a 3-bit address bus. The Reset, Chip Select, Read, and Write pins, along with the Interrupt pin, provide the remaining signals necessary to interface to the CPU. The 82050's system clock can be generated externally and provided through the CLK pin; or its on-chip crystal oscillator can be used by attaching a crystal to the X1 and X2 pins. For compatibility with IBM PC software, a system clock of 18.432 MHz (with divide by two enabled) is recommended. The 82050, along with a transceiver, address decoder, and a crystal, complete the interface to the IBM PC Bus.

SYSTEM CLOCK OPTIONS

The 82050 has two modes of system clock operation. It can accept an externally generated clock, or use a crystal to internally generate its system clock by using the on-chip oscillator.

The 82050 has an on-chip oscillator which can be used to generate its system clock. The oscillator will take the input from a crystal attached to the X1 and X2 pins. The oscillator frequency is divided by two before being inputted into the chip circuitry. If an 18.432 MHz crystal is used, then the actual system clock frequency of the 82050 will be 9.216 MHz. This mode is configured via a strapping option on the $\overline{RTS}$ pin.

CRYSTAL OSCILLATOR



290137-4

Parallel Resonant Crystal Freq. Max = 18.432 MHz (Divided by 2)

Figure 4. Crystal Oscillator

It is very important to distinguish between the clock frequency being supplied into the 82050 and the system clock frequency. The term system clock refers to the clock frequency being supplied to the 82050 circuitry (divided or undivided). The following examples delineate the three options for clock usage and their effect on the 82050 system clock as well as on the BRG source frequency:

1. **Crystal Oscillator:** (Maximum 18.432 MHz)
   - — System Clock Frequency = Crystal Frequency/2
   - — BRG Source Clock Frequency = Crystal Frequency/10
2. **External Clock (Divide by Two Enabled):** (Maximum 18.432 MHz)
   - — System Clock Frequency = External Clock Frequency/2
   - — BRG Source Clock Frequency = External Clock Frequency/10
3. **External Clock (Divide by Two Disabled):** (Maximum 9.216 MHz)
   - — System Clock Freq. = External Clock Frequency
   - — BRG Source Clock Freq. = External Clock Frequency/5

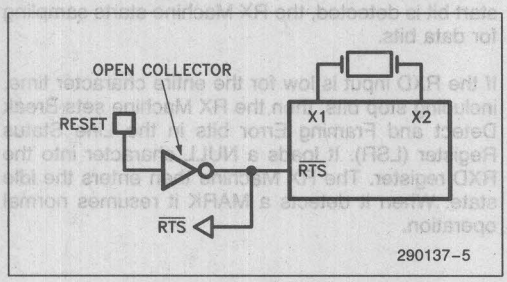


Figure 5. Strapping

During the power up or reset the $\overline{RTS}$ pin is an input; it is weakly pulled high internally and sampled by the falling edge of reset. If it is driven low externally, then the 82050 is configured for a crystal oscillator; otherwise an externally generated clock is expected.

EXTERNALLY GENERATED SYSTEM CLOCK

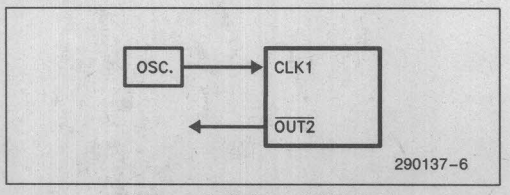


Figure 7. External Clock

This is the default mode of system clock operation. The system clock is divided by two; however, the user may disable the divide by two by a hardware strapping option on the $\overline{DTR}$ pin. The strapping option is similar to the one used on the $\overline{RTS}$ pin.

NOTE:
The use of the Divide by Two strapping option in the crsytal oscillator mode is forbidden.

BAUD RATE GENERATION

The 82050 has a programmable 16-bit Baud Rate Generator (BRG). The 16X baud rate is generated by dividing the source clock with the divisor count from the BRG divisor registers (BAL, BAH). The BRG source clock is the 82050 system clock divided by five. If using an actual 82050 system clock of 9.216 MHz, then the BRG source clock will be 9.216 MHz/5 = 1.8432 MHz, which is compatible with the BRG source clock fed into the IBM PC serial port BRG. This allows the 82050, while using a faster system clock, to maintain full compatibility with software divisor calculations based on the 1.8432 MHz clock used in the IBM PC.

RESET

The 82050 can be reset by asserting the RESET pin. The RESET pin must be held high for at least 8 system clock cycles. If using crystal oscillator, a reset pulse at least 1 ms should be used to ensure oscillator start up. Upon reset, all 82050 registers (except TXD and RXD) are returned to their default states. During reset, the 82050's system clock mode of operation is also selected by strapping options on the RTS and DTR pins (see system clock generation).

INTERRUPTS

The INT pin will go high, or active, whenever one of the following conditions occurs provided it is enabled in the interrupt enable register (IER):

a. Receive Machine Error or Break Condition
b. Receive Data Available
c. Transmit Data Register Empty
d. Change in the State of the Modem Input Pins

The INT pin will be reset (low) when the interrupt source is serviced. The Interrupt Identification Register (IIR) along with the Line Status Register (LSR) and the Modem Status Register (MSR) can be used to identify the source requesting service. The IIR register identifies one of the four conditions listed above. The particular event or status, which triggers the interrupt mechanism, can be identified by reading either the Line Status Register or the Modem Status register. If multiple interrupt sources become active at any one time, then highest priority interrupt source is reflected in the IIR register when the interrupt pin becomes active. Once the highest priority interrupt is serviced, then the next highest priority

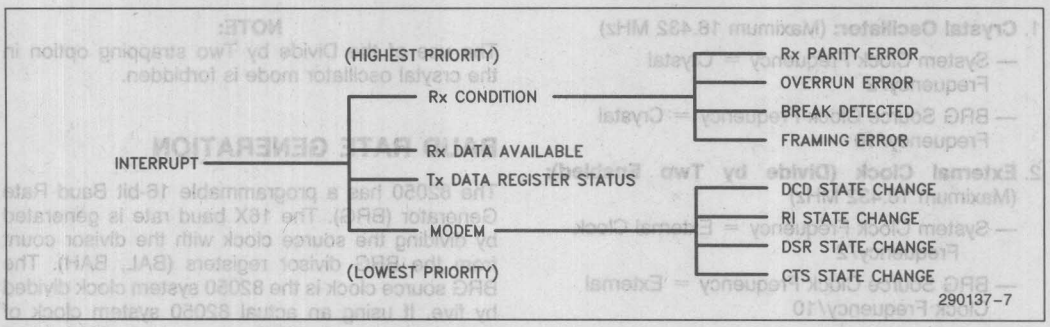


Figure 8. Interrupt Structure

interrupt source is decoded into the IIR register; the whole procedure is repeated until there are no more pending interrupt sources.

TRANSMIT

The 82050 transmission mechanism involves the TX Machine and the TXD Register. The TX Machine reads characters from the TXD Register, serializes the bits, and transmits them over the TXD pin according to signals provided for transmission by the Baud Rate Generator. It also generates parity, and break transmissions upon CPU request.

RECEIVE

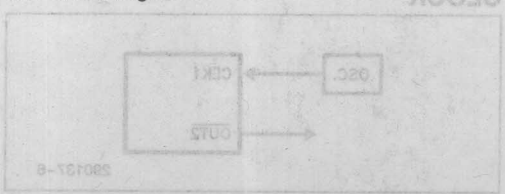
The 82050 reception mechanism involves the RX Machine and the RXD Register. The RX Machine assembles the incoming characters, and loads them onto the RXD Register. The RX Machine synchronizes the data, passes it through a digital filter to filter out spikes, and then uses three samples to generate the bit polarity.

The falling edge of the start bit triggers the RX Machine, which then starts sampling the RXD input (3 samples). If the samples do not indicate a start bit, then a false start bit is determined and the RX Machine returns to the start bit search mode. Once a start bit is detected, the RX Machine starts sampling for data bits.

If the RXD input is low for the entire character time, including stop bits, then the RX Machine sets Break Detect and Framing Error bits in the Line Status Register (LSR). It loads a NULL character into the RXD register. The RX Machine then enters the idle state. When it detects a MARK it resumes normal operation.

SOFTWARE INTERFACE

Like other I/O based peripherals, the 82050 is programmed through its registers to support a variety of functions. The 82050 register set is identical to the 16450 register set to provide compatibility with software written for the IBM PC. The 82050 register set occupies eight addresses and includes control, status, and data registers. The three address lines and the Divisor Latch Access Bit are used to select the 82050 registers.



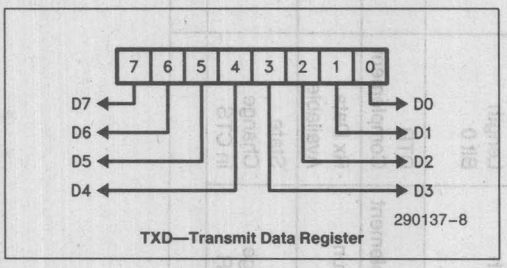
REGISTER DESCRIPTION

Register Map										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
TxD	Tx Data Bit 7	Tx Data Bit 6	Tx Data Bit 5	Tx Data Bit 4	Tx Data Bit 3	Tx Data Bit 2	Tx Data Bit 1	Tx Data Bit 0	0	—
RxD	Rx Data Bit 7	Rx Data Bit 6	Rx Data Bit 5	Rx Data Bit 4	Rx Data Bit 3	Rx Data Bit 2	Rx Data Bit 1	Rx Data Bit 0	0	—
BAL	BRGA LSB Divide Count (DLAB = 1)								0	02H
BAH	BRGA MSB Divide Count (DLAB = 1)								1	00H
IER	0	0	0		Modem Interrupt Enable	Rx Machine Interrupt Enable	Tx Data Interrupt Enable	Rx Data Interrupt Enable	1	00H
IIR	0	0	0	0	0	Active Interrupt Bit 1	Active Interrupt Bit 0	Interrupt Pending	2	01H
LCR	DLAB Divisor Latch Access Bit	Set Break	Parity Mode Bit 2	Parity Mode Bit 1	Parity Mode Bit 0	Stop Bit Length Bit 0	Character Length Bit 1	Character Length Bit 0	3	00H
MCR	0	0	0	Loopback Control Bit	OUT2 Complement	0	RTS Complement	DTR Complement	4	00H
LSR	0	TxM Status	TxD Empty	Break Detected	Framing Error	Parity Error	Overrun Error	Rx Data Available	5	60H
MSR	DCD Input Inverted	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State (H → L) Change in RI	State Change in DSR	State Change in CTS	6	00H
SCR	Scratch-Pad Register								7	00H

Figure 9. Register Description Table

TRANSMIT DATA REGISTER (TXD)

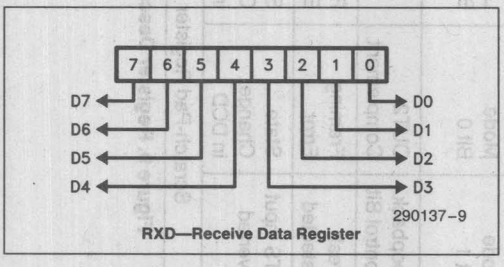
This register holds the next data byte to be transmitted. When the transmit shift register becomes empty, the contents of the Transmit Data Register are loaded into the shift register and the Transmit Data Register Empty condition becomes true.



TXD—Transmit Data Register

RECEIVE DATA REGISTER (RXD)

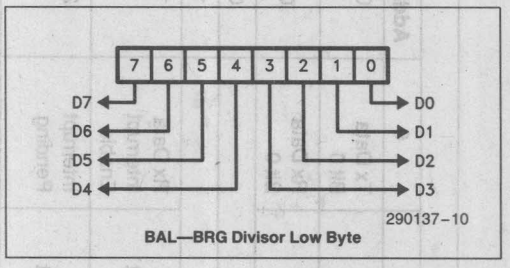
This register holds the last character received by the RX Machine. The character is right justified and the leading bits are zeroed. Reading the register empties the register and resets the Received Character Available condition.



RXD—Receive Data Register

BRG DIVISOR LOW BYTE (BAL)

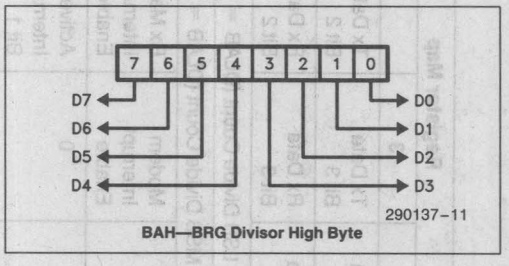
This register contains the least significant byte of the Baud Rate Generator's 16-bit divisor. This register is accessible only when the DLAB bit is set in the LCR register.



BAL—BRG Divisor Low Byte

BRG DIVISOR HIGH BYTE (BAH)

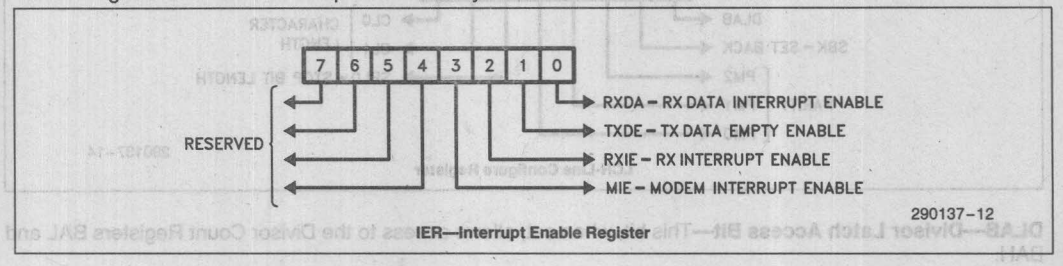
This register contains the most significant byte of the Baud Rate Generator's 16-bit divisor. This register is accessible ony when the DLAB bit is set in the LCR register.



BAH—BRG Divisor High Byte

INTERRUPT ENABLE REGISTER (IER)

This register enables four types of interrupts which independently activate the INT pin. Each of the four interrupt types can be disabled by resetting the appropriate bit of the IER register. Similarly by setting the appropriate bits, selected interrupts can be enabled. If all interrupts are disabled, then the interrupt requests are inhibited from the IIR register and the INT pin. All other functions, including Status Register and the Line Status Register bits continue to operate normally.

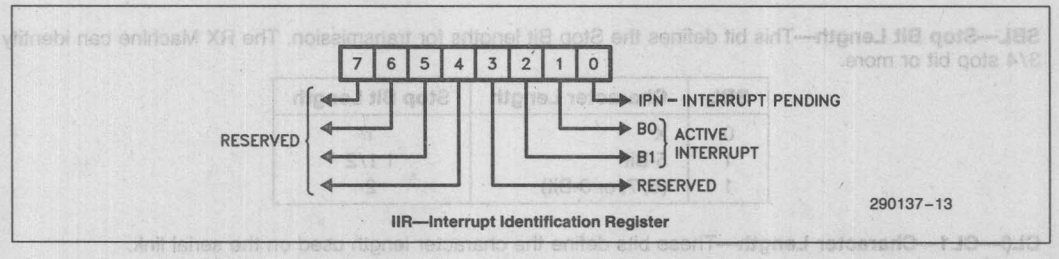


IER—Interrupt Enable Register

MIE—MODEM Interrupt Enable
RXIE—RX Machine Interrupt Enable
TXDE—TX Data Register Empty
RXDA—RX Data Available

INTERRUPT IDENTIFICATION REGISTER (IIR)

This register holds the highest priority enabled and active interrupt request. The source of the interrupt request can be identified by reading bits 2-1.



IIR—Interrupt Identification Register

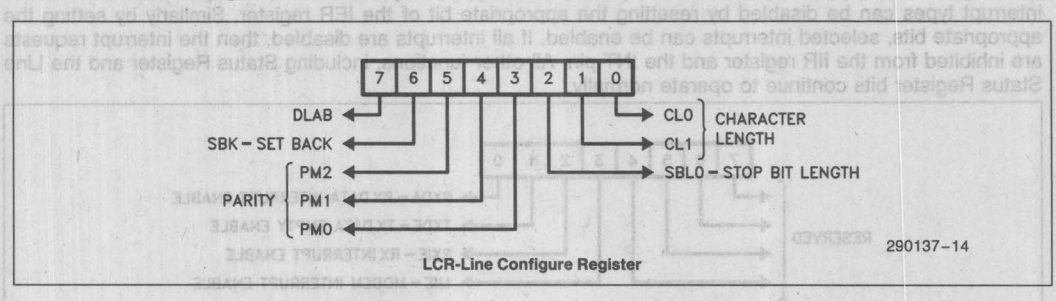
B1, B0—Interrupt Bits, 2-1. These two bits reflect the highest priority, enabled and pending interrupt request.

11: RX Error Condition (Highest Priority)
10: RX Character Available
01: TXD Register Empty
00: Modem Interrupt (Lowest Priority)

IPN—Interrupt Pending—This bit is active low, and indicates that there is an interrupt pending. The interrupt logic asserts the INT pin as soon as this bit goes active (NOTE: the IIR register is continuously updated; so while the user is serving one interrupt source, a new interrupt with higher priority may enter IIR and replace the older interrupt vector).

LINE CONTROL REGISTER (LCR)

This is a read/write register which defines the basic configuration of the serial link.



LCR-Line Configure Register

DLAB—Divisor Latch Access Bit—This bit, when set, allows access to the Divisor Count Registers BAL and BAH.

SBK—Set Break—This will force the TXD pin low. The TXD pin will remain low until this bit is reset.

PM2—PM0—Parity Mode Bits—These three bits are used to select the various parity modes of the 82050.

PM0	PM2	PM1	Function
0	X	X	No Parity
1	0	0	Odd Parity
1	0	1	Even Parity
1	1	0	High Parity
1	1	1	Low Parity

SBL—Stop Bit Length—This bit defines the Stop Bit lengths for transmission. The RX Machine can identify 3/4 stop bit or more.

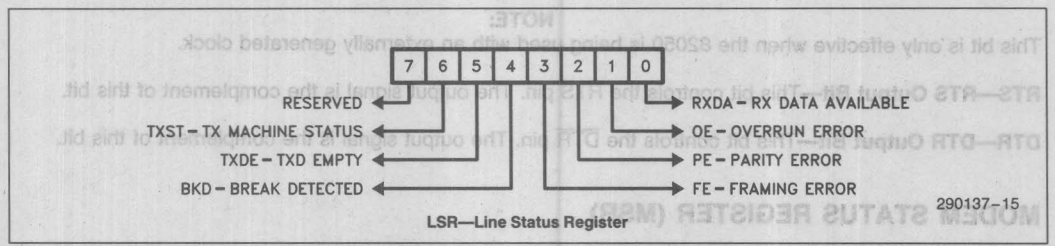
SBL	Character Length	Stop Bit Length
0	X	1
1	5-Bit	1 1/2
1	(6, 7, or 8-Bit)	2

CL0—CL1—Character Length—These bits define the character length used on the serial link.

CL1	CL0	Character Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

LINE STATUS REGISTER (LSR)

This register holds the status of the serial link. When read, all bits of the register are reset to zero.



LSR—Line Status Register

RXDA—RX Data Available—This bit, indicates that the RXD register has data available for the CPU to read.

OE—Overrun Error—Indicates that a received character was lost because the RXD register was not empty.

PE Parity Error—Indicates that a received character had a parity error.

FE—Framing Error—Indicates that a received character had a framing error.

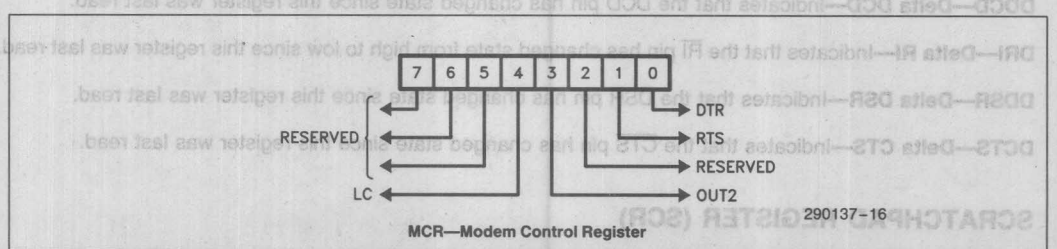
BkD—Break Detected—This bit indicates that a break condition was detected, i.e., RxD input was held low for two character times.

TXDE—TXD Empty—This indicates that the 82050 is ready to accept a new character for transmission. In addition, this bit causes an interrupt request to be generated if the TXD register Empty interrupt is enabled.

TXST—TX Machine Status—When set, this bit indicates that the TX Machine is Empty, i.e., both the TXD register and the TX Shift Register are empty.

MODEM CONTROL REGISTER (MCR)

This register controls the modem output pins. All the outputs invert the data, i.e., their output will be the complement of the data written into this register.



MCR—Modem Control Register

LC—Loopback Control—This bit puts the 82050 into a Local Loopback mode.

OUT2—OUT2 Output—This bit controls the $\overline{OUT2}$ pin. The output signal is the complement of this bit.

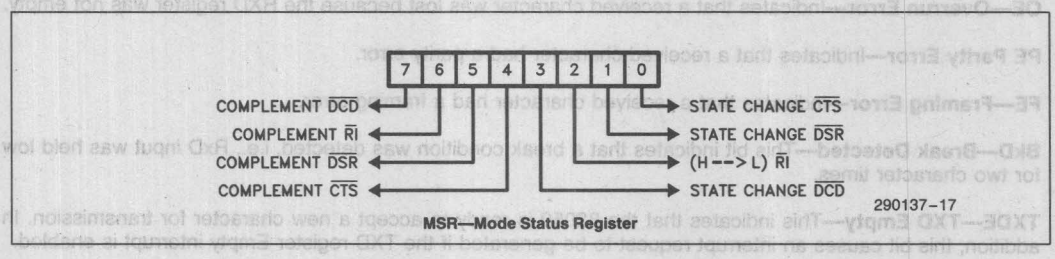
NOTE:
This bit is only effective when the 82050 is being used with an externally generated clock.

RTS—RTS Output Bit—This bit controls the $\overline{RTS}$ pin. The output signal is the complement of this bit.

DTR—DTR Output Bit—This bit controls the $\overline{DTR}$ pin. The output signal is the complement of this bit.

MODEM STATUS REGISTER (MSR)

This register holds the status of the modem input pins (CTS, DCD, DSR, RI). It is the source of Modem interrupts (bits 3–0) when enabled in the IER register. If any of the above input pins change levels, then the appropriate bit in MSR is set. Reading MSR will clear the status bits.



MSR—Mode Status Register

DCDC—DCD Complement—Holds the complement of the $\overline{DCD}$ pin.

DRIC—RI Complement—Holds the complement of the $\overline{RI}$ pin.

DSRC—DSR Complement—Holds the complement of the $\overline{DSR}$ pin.

CTSC—CTS Complement—Holds the complement of the $\overline{CTS}$ pin.

DDCD—Delta DCD—Indicates that the $\overline{DCD}$ pin has changed state since this register was last read.

DRI—Delta RI—Indicates that the $\overline{RI}$ pin has changed state from high to low since this register was last read.

DDSR—Delta DSR—Indicates that the $\overline{DSR}$ pin has changed state since this register was last read.

DCTS—Delta CTS—Indicates that the $\overline{CTS}$ pin has changed state since this register was last read.

SCRATCHPAD REGISTER (SCR)

The 8-bit Read/Write register does not control the ACC. It is intended as a scratch pad register for use by the programmer.

V_{IH}	Input High Voltage	(1), (7)	2.0	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	(2), (9)		0.45	V
V_{OH}	Output High Voltage	(3), (9)	2.4		V
I_{LI}	Input Leakage Current	(4)		±10	μA
I_{LO}	3-State Leakage Current	(5)		±10	μA
I_{OHR}	Input High for $\overline{DTR}$, $\overline{RTS}$	(10)		0.4	mA
I_{OLR}	Input Low for $\overline{DTR}$, $\overline{RTS}$	(10)	11		mA
L_{XTAL}	X1, X2 Load			10	pF
I_{CC}	Power Supply Current	(6)		3.8 35	mA/MHz mA (max)
C_{in}	Input Capacitance	(8)		10	pF
C_{io}	I/O Capacitance	(8)		10	pF
C_{XTAL}	X_1, X_2 Load			10	pF

NOTES:
1. Does not apply to CLK/X1 pin, when configured as crystal oscillator input (X1).
2. @ I_{ol} = 2 mA.
3. @ I_{oh} = −0.4 mA.
4. $0 < V_{in} < V_{CC}$.
5. $0.45V < V_{out} < (V_{CC} - 0.45)$.
6. V_{CC} = 5.5V; V_{il} = 0.5V (max); $V_{ih} = V_{CC} - 0.5V$ (min); $I_{ol} = I_{oh} = 0$; 9.2 MHz (max).
7. V_{IH} = 2.4V on $\overline{RD}$ and RXD pins.
8. Freq = 1 MHz.
9. Does not apply OUT2/X2 pin, when configured as crystal oscillator output (X2).
10. Input current for $\overline{DTR}$, $\overline{RTS}$ pins during Reset for Clock Mode Configuration.

A.C. SPECIFICATIONS

Testing Conditions:

- All AC output parameters are under output load of 20 to 100 pF, unless otherwise specified.
- AC testing inputs are driven at 2.4 for logic '1', and 0.45V for logic '0'. Output timing measurements are made at 1.5V for both a logical '0' and '1'.
- In the following tables, the units are ns, unless otherwise specified.

System Interface Specification—System Clock Specification:

The 82050 system clock is supplied via the CLK pin or generated by on-chip crystal oscillator. The clock is optionally divided by two. The CLK parameters are given separately for internal divide-by-two option ACTIVE and INACTIVE.

The system clock (after division by two, if active) must be at least 16X the Tx or Rx baud rate (the faster of the two).

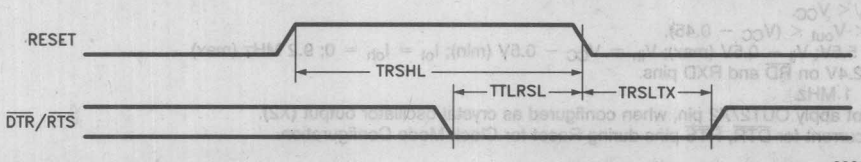
SYSTEM CLOCK SPECIFICATIONS

Symbol	Parameter	Min	Max	Notes
DIVIDE BY TWO OPTION—ACTIVE				
Tcy/2	CLK Period	54	250	(2)
TCLCH	CLK Low Time	25		
TCHCL	CLK High Time	25		
TCH1CH2	CLK Rise Time		10	(1)
TCL2CL1	CLK Fall Time		10	(1)
FXTAL	External Crystal Frequency Rating	4.0	18.432 MHz	
DIVIDE BY TWO OPTION—INACTIVE				
Tcy	CLK Period	108		
TCLCH	CLK Low Time	54		
TCHCL	CLK High Time	44	250	
TCH1CH2	CLK Rise Time		15	(1)
TCL2CL1	CLK Fall Time		15	(1)

NOTES:
1. Rise/fall times are measured between 0.8 and 2.0V.
2. Tcy in ACTIVE divide by two option is TWICE the input clock period.

RESET SPECIFICATION

Symbol	Parameter	Min	Max	Notes
TRSHL	Reset Width—CLK/X1 Configured to CLK	8 Tcy		(1)
TTLRSL	RTS/DTR LOW Setup to Reset Inactive	6 Tcy		(2)
TRSLTX	RTS/DTR Low Hold after Reset Inactive	0	Tcy − 20	(2)



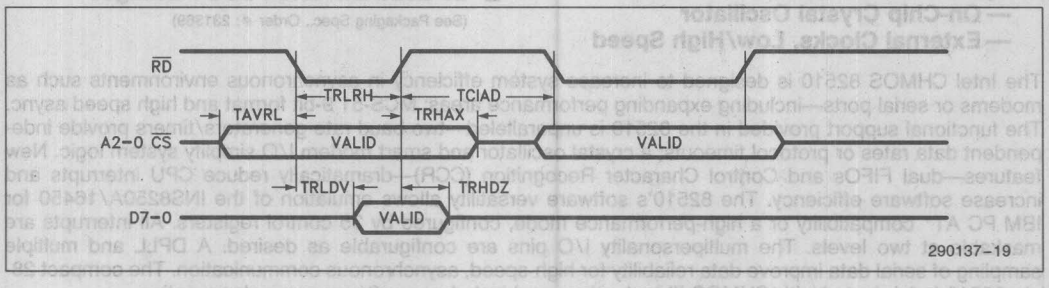
290137-18

NOTES:
1. In case of CLK/X1 configured as X1, additional time is required to guarantee crystal oscillator wake-up.
2. RTS/DTR are internally driven HIGH during RESET active time. The pin should be either left OPEN or externally driven LOW during RESET according to the required configuration of the system clock. These parameters specify the timing requirements on these pins, in case they are externally driven LOW during RESET. The maximum spec on TRSLTX requires that the RTS/DTR pins not be forced later than TRSLTX maximum.

READ CYCLE SPECIFICATIONS

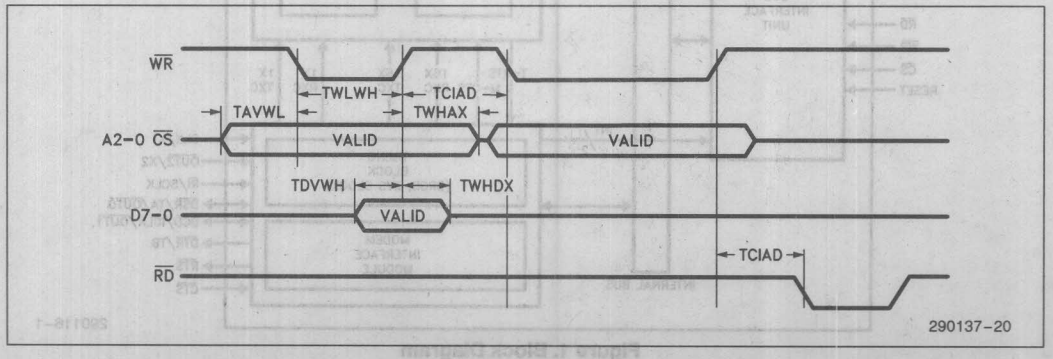
Symbol	Parameter	Min	Max	Notes
TRLRH	$\overline{RD}$ Active Width	2 Tcy + 65		
TAVRL	Address/$\overline{CS}$ Setup Time to $\overline{RD}$ Active	7		
TRHAX	Address/$\overline{CS}$ Hold Time after $\overline{RD}$ Inactive	0		
TRLDV	Data Out Valid after $\overline{RD}$ Active		2Tcy + 65	(1)
TCIAD	Command Inactive to Active Delay	Tcy + 15		(2)
TRHDZ	Data Out Float Delay after $\overline{RD}$ Inactive		40	

NOTES:
1. C1 = 20 pF to 100 pF.
2. Command refers to either Read or Write signals.



WRITE CYCLE SPECIFICATION

Symbol	Parameter	Min	Max	Notes
TWLWH	$\overline{WR}$ Active Width	2Tcy + 15		
TAVWL	Address $\overline{CS}$ Setup Time to $\overline{WR}$ Active	7		
TWHAX	Address and $\overline{CS}$ Hold Time after $\overline{WR}$	0		
TDVWH	Data in Setup Time to $\overline{WR}$ Inactive	90		
TWHDX	Data in Hold Time after $\overline{WR}$ Inactive	12		



82510
ASYNCHRONOUS SERIAL CONTROLLER

- **Asynchronous Operation**
  - **— 5- to 9-Bit Character Format**
  - **— Baud Rate DC to 288k**
  - **— Complete Error Detection**
- **Multiple Sampling Windows**
- **Two, Independent, Four-Byte Transmit and Receive FIFOs with Programmable Threshold**
- **Two, 16-Bit Baud Rate Generators/ Timers**
- **System Clock Options**
  - **— On-Chip Crystal Oscillator**
  - **— External Clocks, Low/High Speed**
- **MCS-51 9-Bit Protocol Support**
- **IBM PC AT* (INS 8250A/16450®) Software Compatible**
- **Control Character Recognition**
- **CHMOS III with Power Down Mode**
- **Interrupts Maskable at Two Levels**
- **Auto Echo and Loopback Modes**
- **Seven I/O Pins, Dedicated and General Purpose**
- **28-Lead DIP and PLCC Packages**

(See Packaging Spec., Order #: 231369)

The Intel CHMOS 82510 is designed to increase system efficiency in asynchronous environments such as modems or serial ports—including expanding performance areas: MCS-51 9-bit format and high speed async. The functional support provided in the 82510 is unparalleled—two baud rate generators/timers provide independent data rates or protocol timeouts; a crystal oscillator and smart modem I/O simplify system logic. New features—dual FIFOs and Control Character Recognition (CCR)—dramatically reduce CPU interrupts and increase software efficiency. The 82510's software versatility allows emulation of the INS8250A/16450 for IBM PC AT* compatibility or a high-performance mode, configured by 35 control registers. All interrupts are maskable at two levels. The multipersonality I/O pins are configurable as desired. A DPLL and multiple sampling of serial data improve data reliability for high-speed, asynchronous communication. The compact 28-pin 82510 is fabricated with CHMOS III technology and includes a software powerdown option.

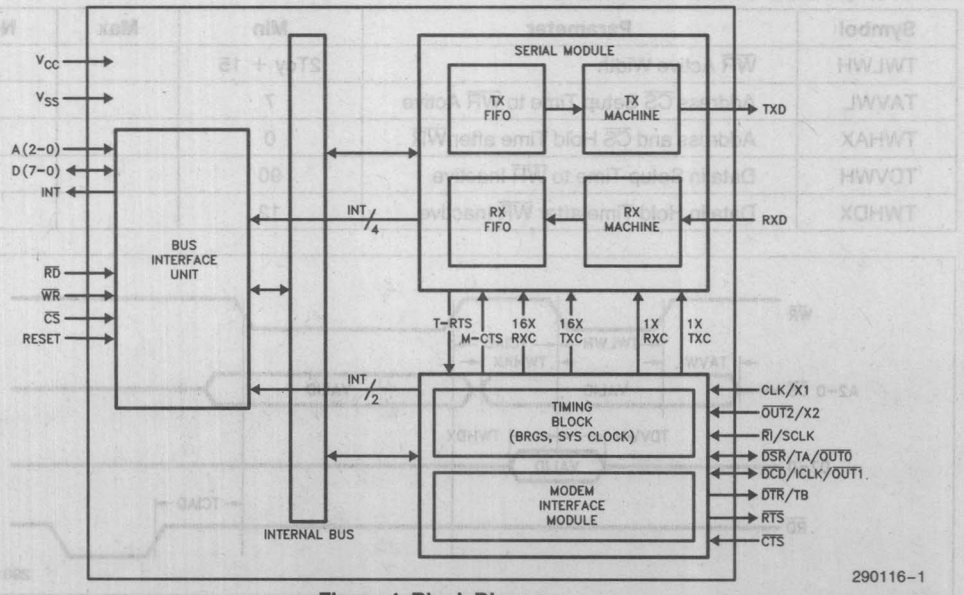


Figure 1. Block Diagram

*IBM and PC AT are trademarks of International Business Machines.

October 1989
Order Number: 290116-004

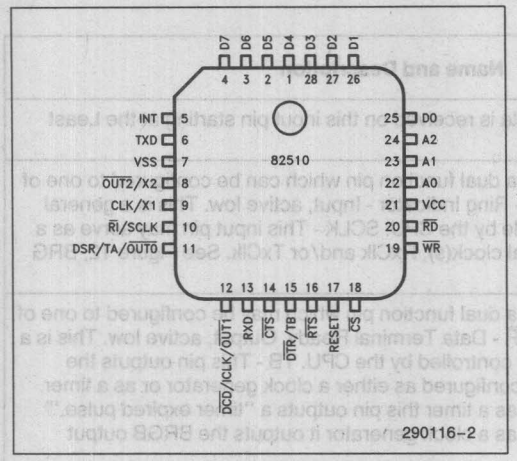


Figure 2. PLCC Pinout

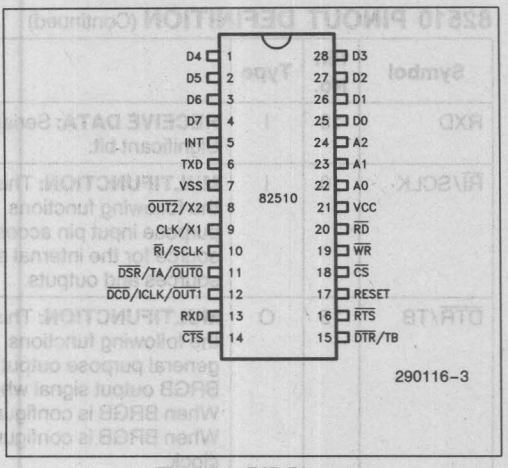


Figure 3. DIP Pinout

2

82510 PINOUT DEFINITION

Symbol	Pin No.	Type	Name and Description
RESET	17	I	**RESET:** A high on this input pin resets the 82510 to the Default Wake-up mode.
$\overline{CS}$	18	I	**CHIP SELECT:** A low on this input pin enables the 82510 and allows read or write operations.
A2–A0	24-22	I	**ADDRESS PINS:** These inputs interface with three bits of the System Address Bus to select one of the internal registers for read or write.
D7–D0	4* 25	I/O	**DATA BUS:** Bi-directional, three state, eight-bit Data Bus. These pins allow transfer of bytes between the CPU and the 82510.
$\overline{RD}$	20	I	**READ:** A low on this input pin allows the CPU to read Data or Status bytes from the 82510.
$\overline{WR}$	19	I	**WRITE:** A low on this input allows the CPU to write Data or Control bytes to the 82510.
INT	5	O	**INTERRUPT:** A high on this output pin signals an interrupt request to the CPU. The CPU may determine the particular source and cause of the interrupt by reading the 82510 Status registers.
CLK/X1	9	I	**MULTIFUNCTION:** This input pin serves as a source for the internal system clock. The clock may be asynchronous to the serial clocks and to the processor clock. This pin may be used in one of two modes: CLK — in this mode an externally generated TTL compatible clock should be used to drive this input pin; X1 — in this mode the clock is internally generated by an on-chip crystal oscillator. This mode requires a crystal to be connected between this pin (X1) and the X2 pin. (See System Clock Generation.)
$\overline{OUT2}$/X2	8	O	**MULTIFUNCTION:** This is a dual function pin which may be configured to one of the following functions: $\overline{OUT2}$ — a general purpose output pin controlled by the CPU, only available when CLK/X1 pin is driven by an externally generated clock; X2 - this pin serves as an output pin for the crystal oscillator. *Note*: The configuration of the pin is done only during hardware reset. For more details refer to the System Clock Generation.
TXD	6	O	**TRANSMIT DATA:** Serial data is transmitted via this output pin starting at the Least Significant bit.

*Pins 28–25 and Pins 4–1.

82510 PINOUT DEFINITION (Continued)

Symbol	Pin No.	Type	Name and Description
RXD	13	I	**RECEIVE DATA:** Serial data is received on this input pin starting at the Least Significant bit.
$\overline{RI}$/SCLK	10	I	**MULTIFUNCTION:** This is a dual function pin which can be configured to one of the following functions. $\overline{RI}$ - Ring Indicator - Input, active low. This is a general purpose input pin accessible by the CPU. SCLK - This input pin may serve as a source for the internal serial clock(s), RxClk and/or TxClk. See Figure 12, BRG sources and outputs.
$\overline{DTR}$/TB	15	O	**MULTIFUNCTION:** This is a dual function pin which may be configured to one of the following functions. $\overline{DTR}$ - Data Terminal Ready. Output, active low. This is a general purpose output pin controlled by the CPU. TB - This pin outputs the BRGB output signal when configured as either a clock generator or as a timer. When BRGB is configured as a timer this pin outputs a "timer expired pulse." When BRGB is configured as a clock generator it outputs the BRGB output clock.
$\overline{DSR}$/TA/ $\overline{OUT0}$	11	I/O	**MULTIFUNCTION:** This is a multifunction pin which may be configured to one of the following functions. $\overline{DSR}$ - Data Set Ready. Input, active low. This is a general purpose input pin accessible by the CPU. TA - This pin is similar in function to pin TB except it outputs the signals from BRGA instead of BRGB. $\overline{OUT0}$ - Output pin. This is a general purpose output pin controlled by the CPU.
$\overline{RTS}$	16	O	**REQUEST TO SEND:** Output pin, active low. This is a general purpose output pin controlled by the CPU. In addition, in automatic transmission mode this pin, along with $\overline{CTS}$, controls the transmission of data. (See Transmit modes for further detail.) During hardware reset this pin is an input. It is used to determine the System Clock Mode. (See System Clock Generation for further detail.)
$\overline{CTS}$	14	I	**CLEAR TO SEND:** Input pin, active low. In automatic transmission mode it directly controls the Transmit Machine. (See transmission mode for further details.) This pin can be used as a General Purpose Input.
$\overline{DCD}$/ICLK/ $\overline{OUT1}$	12	I/O	**MULTIFUNCTION:** This is a multifunction pin which may be configured to one of the following functions. $\overline{DCD}$ - Data Carrier Detected. Input pin, active low. This is a general purpose input pin accessible by the CPU. ICLK - This pin is the output of the internal system clock. $\overline{OUT1}$ - General purpose output pin. Controlled by the CPU.
V_{CC}	21	P	**V_{CC}:** Device power supply.
V_{SS}	7	P	**V_{SS}:** Ground.

Table 1. Multifunction Pins

Pin #	I/O	Timing	Modem
8	*$\overline{OUT2}$	X2	—
9	—	*CLK/X1	—
10	—	SCLK	*$\overline{RI}$
11	$\overline{OUT0}$	TA	*$\overline{DSR}$
12	$\overline{OUT1}$	ICLK	*$\overline{DCD}$
14	—	—	*$\overline{CTS}$
15	—	TB	*$\overline{DTR}$
16	—	—	*$\overline{RTS}$

*Default

GENERAL DESCRIPTION

The 82510 can be functionally divided into seven major blocks (See Fig 1): Bus Interface Unit, Timing Unit, Modem Module, Tx FIFO, Rx FIFO, Tx Machine, and Rx Machine. Six of these blocks (all except Bus Interface Unit) can generate block interrupts. Three of these blocks can generate second-level interrupts which reflect errors/status within the block (Receive Machine, Timing Unit, and the Modem Module).

The Bus interface unit allows the 82510 to interface with the rest of the system. It controls access to device registers as well as generation of interrupts to the external world. The FIFOs buffer the CPU from the Serial Machines and reduce the interrupt overhead normally required for serial operations. The threshold (level of occupancy in the FIFO which will generate an interrupt) is programmable for each FIFO. The timing unit controls generation of the system clock through either its on-chip crystal oscillator, or an externally generated clock. It also provides two Baud Rate Generators/Timers with various options and modes to support serial communication.

FUNCTIONAL DESCRIPTION

CPU Interface

The 82510 has a simple demultiplexed Bus Interface, which consists of a bidirectional three-state eight-bit, data bus and a three-bit address bus. An Interrupt pin along with the Read, Write and Chip Select are the remaining signals used to interface with the CPU. The three address lines along with the Bank Pointer register are used to select the registers. The 82510 is designed to interface to all Intel microprocessor and microcontroller families. Like most other I/O based peripherals it is programmed through its registers to support a variety of functions. Its register set can be used in 8250A/16450 compatibility or High Performance modes. The 8250A/16450 mode is the default wake-up mode in which only the 8250A/16450 compatible registers are accessible. The remaining registers are default configured to support 8250A/16450 emulation.

Software Interface

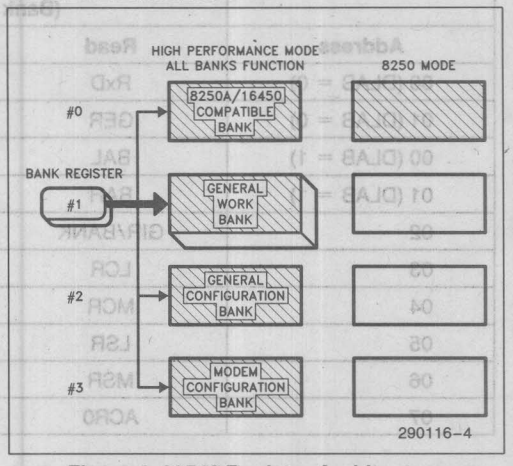


Figure 4. 82510 Register Architecture

The 82510 is configured and controlled through its 35 registers which are divided into four banks. Only one bank is accessible at any one time. The bank switching is done by changing the contents of the bank pointer (GIR/BANK–BANK0, BANK1). The banks are logically grouped into 8250A/16450 compatible (0), General Work Bank (1), General Configuration (2), and Modem Configuration (3). The 8250A/16450 compatible bank (Bank 0) is the default bank upon power up.

The 82510 registers can be categorized under the following:

Table 2. 82510 Register/Block Functions

	Status	Enable	Configuration	Command	Data
FIFO	FLR	—	FMD	—	—
MODEM	MSR	MIE	PMD	MCR	—
RX	RST, RXF	RIE	RMD	RCM	RXD, RXF
TX	LSR	LSR	TMD	TCM	TXD, TXF
TIMER	TMST	TMIE	CLCF, BACF, BBCF	TMCR	BBL, BBH BAL, BAH
DEVICE	GSR, GIR	GER	IMD	ICM	—
8250	LSR, MSR, GIR	GER	LCR, MCR	MCR	TXD, RXD BAL, BAH

8250 Compatibility

Upon power up or reset, the 82510 comes up in the default wake up mode. The 8250A/16450 compatible bank, bank zero, is the accessible bank and all the other registers are configured via their default values to support this mode. An 18.432 MHz crystal frequency is necessary.

Table 3. 8250A/16450 Compatible Registers

	82510 Registers (Bank 0)		8250A Registers	
Address	**Read**	**Write**	**Read**	**Write**
00 (DLAB = 0)	RxD	TxD	RBR	THR
01 (DLAB = 0)	GER	GER	IER	IER
00 (DLAB = 1)	BAL	BAL	DLL	DLL
01 (DLAB = 1)	BAH	BAH	DLM	DLM
02	GIR/BANK	BANK	IIR	—
03	LCR	LCR	LCR	LCR
04	MCR	MCR	MCR	MCR
05	LSR	LSR	LSR	LSR
06	MSR	MSR	MSR	MSR
07	ACR0	ACR0	SCR	SCR

Table 4. Default Wake-Up Mode

RxD	—	ACR1	00H	RxF	—
TxD	—	RIE	1EH	TxF	—
BAL	02H	RMD	00H	TMST	30H
BAH	00H	CLCF	00H	TMCR	—
GER	00H	BACF	04H	FLR	00H
GIR/BANK	01H	BBCF	84H	RCM	—
LCR	00H	PMD	FCH	TCM	—
MCR	00H	MIE	0FH	GSR	12H
LSR	60H	TMIE	00H	ICM	—
MSR	00H	BBL	05H	FMD	00H
ACR0	00H	BBH	00H	TMD	00H
RST	00H			IMD	0CH

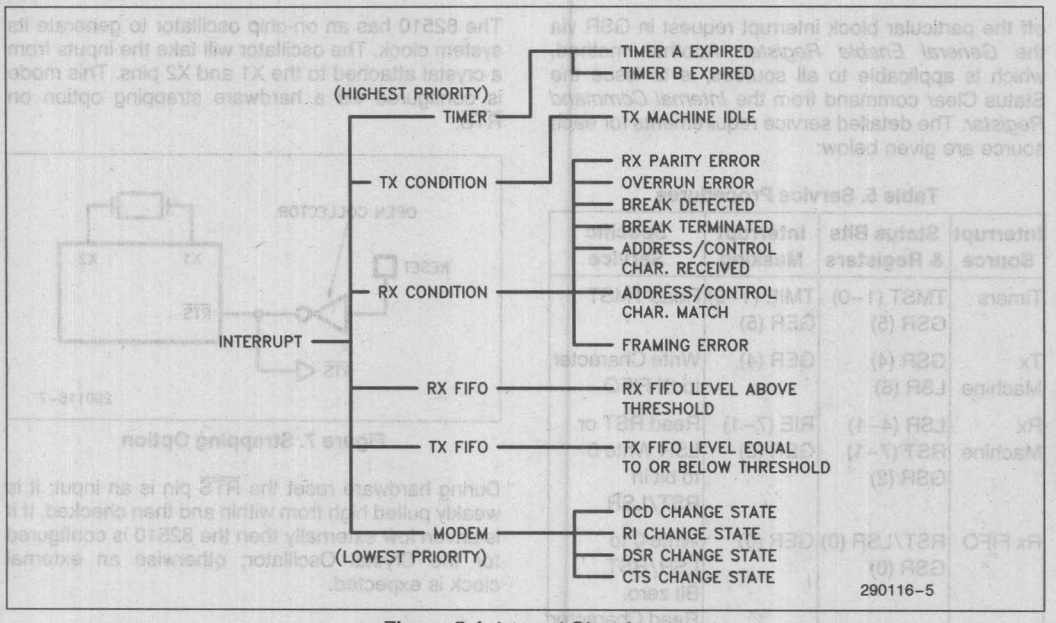


Figure 5. Interrupt Structure

Interrupts

There are two levels of interrupt/status reporting within the 82510. The first level is the block level interrupts such as RX FIFO, Tx FIFO, Rx Machine, Tx Machine, Timing unit, and Modem Module. The status of these blocks is reported in the General Status and General Interrupt Registers. The second level is the various sources within each block; only three of the blocks generate second level interrupts (Rx Machine, Timing Unit, and Modem Module). Interrupt requests are maskable at both the block level and at the individual source level within the module. If more than one unmasked block requests interrupt service an on-chip interrupt controller will resolve contention on a priority basis (each block has a fixed priority). An interrupt request from a particular block is activated if one of the unmasked status bits within the status register for the block is set. A CPU service operation, e.g., reading the appropriate status register, will reset the status bits.

ACKNOWLEDGE MODES

The interrupt logic will assert the INT pin when an interrupt is coded into the General Interrupt register. The INT pin is forced low upon acknowledgment. The 82510 has two modes of interrupt acknowledgment:

1. Manual Acknowledge

The CPU must issue an explicit Interrupt Acknowledge command via the Interrupt Acknowledge bit of the Internal Command register. As a result the INT pin is forced low for two clocks and then updated.

2. Automatic Acknowledge

As opposed to the Manual Acknowledge mode, when the CPU must issue an explicit interrupt acknowledge command, an interrupt service operation is considered as an automatic acknowledgment. This forces the INT pin low for two clock cycles. After two cycles the INT pin is updated, i.e., if there is still an active non-masked interrupt request the INT pin is set HIGH.

INTERRUPT SERVICE

A service operation is an operation performed by the CPU, which causes the source of the 82510 interrupt to be reset (it will reset the particular status bit causing the interrupt). An interrupt request within the 82510 will not reset until the interrupt source has been serviced. Each source can be serviced in two or three different ways; one general way is to disable the particular status bit causing the interrupt, via the corresponding block enable register. Setting the appropriate bit of the enable register to zero will mask off the corresponding bit in the status register, thus causing an edge on the input line to the interrupt logic. The same effect can be achieved by masking

off the particular block interrupt request in GSR via the *General Enable Register.* Another method, which is applicable to all sources, is to issue the Status Clear command from the *Internal Command Register.* The detailed service requirements for each source are given below:

Table 5. Service Procedures

Interrupt Source	Status Bits & Registers	Interrupt Masking	Specific Service
Timers	TMST (1–0) GSR (5)	TMIE (1–0) GER (5)	Read TMST
Tx Machine	GSR (4) LSR (6)	GER (4)	Write Character to tX FIFO
Rx Machine	LSR (4–1) RST (7–1) GSR (2)	RIE (7–1) GER (2)	Read RST or LSR Write 0 to bit in RST/LSR
Rx FIFO	RST/LSR (0) GSR (0)	GER (0)	Write 0 to LSR/RST Bit zero. Read Character
Tx FIFO	LSR (5) GSR (1)	GER (1)	Write to FIFO Read GIR(1)
Modem	MSR (3-0) GSR (3)	MIE (3-0) GER (3)	Read MSR write 0 into the appropriate bits of MSR (3–0).

NOTE:
1. Only if pending interrupt is Tx FIFO.

System Clock Generation

The 82510 has two modes of System Clock Operation. It can accept an externally generated clock, or it can use a crystal to internally generate its system clock.

CRYSTAL OSCILLATOR

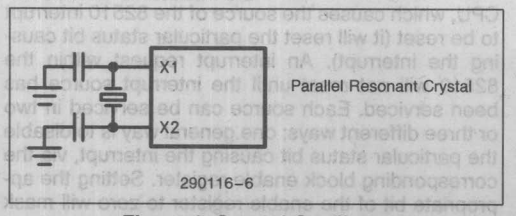


Figure 6. Crystal Oscillator

The 82510 has an on-chip oscillator to generate its system clock. The oscillator will take the inputs from a crystal attached to the X1 and X2 pins. This mode is configured via a hardware strapping option on $\overline{RTS}$.

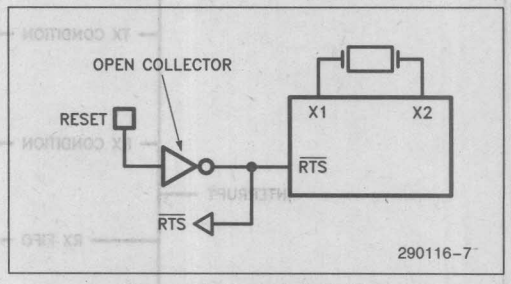


Figure 7. Strapping Option

During hardware reset the $\overline{RTS}$ pin is an input; it is weakly pulled high from within and then checked. If it is driven low externally then the 82510 is configured for the Crystal Oscillator; otherwise an external clock is expected.

EXTERNALLY GENERATED SYSTEM CLOCK

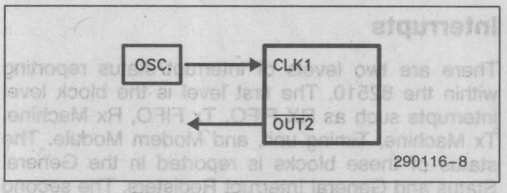


Figure 8. External Clock

This is the default configuration. Under normal conditions the system clock is divided by two; however, the user may disable divide by two via a hardware strapping option on the $\overline{DTR}$ pin. The Hardware strapping option is similar to the one used on the $\overline{RTS}$ pin. It is forbidden to strap both $\overline{DTR}$ and $\overline{RTS}$.

Transmit

The two major blocks involved in transmission are the Transmit FIFO and the Transmit Machine. The Tx FIFO acts as a buffer between the CPU and the Tx Machine. Whenever a data character is written to the Transmit Data register, it, along with the Transmit Flags (if applicable), is loaded into the Tx FIFO.

TX FIFO

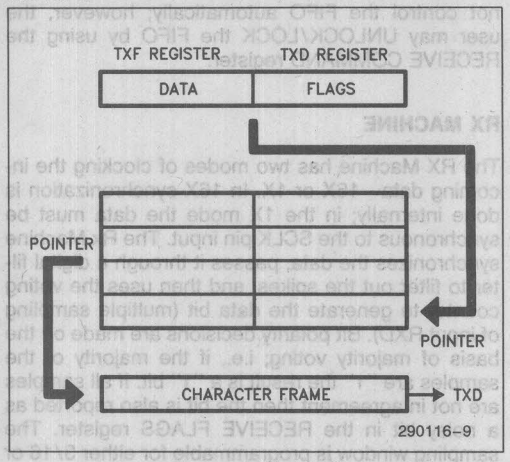


Figure 9. Tx FIFO

The Tx FIFO can hold up to four, eleven-bit characters (nine-bits data, parity, and address flag). It has separate read and write mechanisms. The read and write pointers are incremented after every operation to allow data transfer to occur in a First In First Out fashion. The Tx FIFO will generate a maskable interrupt when the level in the FIFO is below, or equal to, the Threshold. The threshold is user programmable.

For example, if the threshold equals two, and the number of characters in the Tx FIFO decreases from three to two, the FIFO will generate an interrupt. The threshold should be selected with regard to the system's interrupt service latency.

NOTE:
There is a one character transmission delay between FIFO empty and Transmitter Idle, so a threshold of zero may be selected without getting an underrun condition. Also if more than four characters are written to the FIFO an overrun will occur and the extra character will not be written to the Tx FIFO. This error will not be reported to the CPU.

TX MACHINE

The Tx Machine reads characters from the Tx FIFO, serializes the bits, and transmits them over the TXD pin according to the timing signals provided for transmission. It will also generate parity, transmit break (upon CPU request), and manage the modem handshaking signals ($\overline{CTS}$ and $\overline{RTS}$) if configured so. The Tx machine can be enabled or disabled through the Transmit Command register or $\overline{CTS}$. If the transmitter is disabled in the middle of a character transmission the transmission will continue until the end of the character; only then will it enter the disable state.

TRANSMIT CLOCKS

There are two modes of transmission clocking, 1X and 16X. In the 1X mode the transmitted data is synchronous to the transmit clock as supplied by the SCLK pin. In this mode stop-bit length is restricted to one or two bits only. In the 16X mode the data is not required to be synchronous to the clock. (Note: The Tx clock can be generated by the BRGs or from the SCLK pin.)

MODEM HANDSHAKING

The transmitter has three modes of handshaking.

Manual Mode—In this mode the $\overline{CTS}$ and $\overline{RTS}$ pins are not used by the Tx Machine (transmission is started regardless of the $\overline{CTS}$ state, and $\overline{RTS}$ is not forced low). The CPU may manage the handshake itself, by accessing the $\overline{CTS}$ and $\overline{RTS}$ signals through the MODEM CONTROL and MODEM STATUS registers.

Semi-Automatic Mode—In this mode the $\overline{RTS}$ pin is activated whenever the transmitter is enabled. The $\overline{CTS}$ pin's state controls transmission. Transmission is enabled only if $\overline{CTS}$ is active. If $\overline{CTS}$ becomes inactive during transmission, the Tx Machine will complete transmission of the current character and then go to the inactive state until $\overline{CTS}$ becomes active again.

Automatic Mode—This mode is similar to the semi-automatic mode, except that $\overline{RTS}$ will be activated as long as the transmitter is enabled and there are more characters to transmit. The CPU need only fill the FIFO, the handshake is done by the Tx Machine. When both the shift register and the FIFO are empty $\overline{RTS}$ automatically goes inactive. (Note: The $\overline{RTS}$ pin can be forced to the active state by the CPU, regardless of the handshaking mode, via the MODEM CONTROL register.)

Receive

The 82510 reception mechanism involves two major blocks; the Rx Machine and the Rx FIFO. The Rx Machine will assemble the incoming character and its associated flags and then LOAD them on to the Rx FIFO. The top of the FIFO may be read by reading the Receive Data register and the Receive Flags Register. The receive operation can be done in two modes. In the *normal* mode the characters are received in the standard Asynchronous format and only control characters are recognized. In the *ulan* mode, the nine bit protocol of the MCS-51 family is supported and the ulan Address characters, rather than Control Characters are recognized.

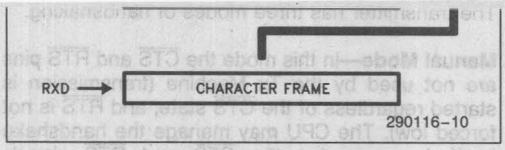


Figure 10. Rx FIFO

The Rx FIFO is very similar in structure and basic operation to the Tx FIFO. It will generate a maskable interrupt when the FIFO level is above the threshold. The Rx FIFO can also be configured to operate as a one-byte buffer. This mode is used for 8250 compatible software drivers. An overrun will occur when the FIFO is full and the Rx Machine has a new character for the FIFO. In this situation the oldest character is discarded and the new character is loaded from the Rx Machine. An Overrun error bit will also be set in the RECEIVE STATUS and LINE STATUS registers.

The user has the option to disable the loading of incoming characters on to the Rx FIFO by using the UNLOCK/LOCK FIFO commands. (See RECEIVE COMMAND register.) When the Rx FIFO is locked, it will ignore load requests from the Rx Machine, and thus the received characters will not be loaded into the FIFO and may be lost (if another character is received). These two commands are useful when the CPU is not willing to receive characters, or is waiting for specific Control/Address characters. In uLAN mode there are three options of address recognition, each of these options varies in the amount of CPU offload, and degree of FIFO control through OPEN/LOCK FIFO commands.

Automatic Mode—In this mode the Rx Machine will open the FIFO whenever an Address Match occurs; it will LOCK the FIFO if an address mismatch occurs.

Semi-Automatic Mode—In this mode the Rx Machine will open the FIFO whenever an address character is received. It will not lock the FIFO if the Address does not match. The user is responsible for locking the Rx FIFO.

counter to generate the data bit (multiple sampling of input RXD). Bit polarity decisions are made on the basis of majority voting; i.e., if the majority of the samples are "1" the result is a "1" bit. If all samples are not in agreement then the bit is also reported as a noisy bit in the RECEIVE FLAGS register. The sampling window is programmable for either 3/16 or 7/16 samples. The 3/16 mode is useful for high frequency transmissions, or when serious RC delays are expected on the channel. The 7/16 is best suited for noisy media. The Rx machine also has a DPLL to overcome frequency shift problems; however, using it in a very noisy environment may increase the error, so the user can disable the DPLL via the Receive Mode register. The Rx Machine will generate the parity and the address marker as well as any framing error indications.

Start Bit Detection—The falling edge of the Start bit resets the DPLL counter and the Rx Machine starts sampling the input line (the number of samples is determined by the configuration of the sampling window mode). The Start bit verification can be done through either a majority voting system or an absolute voting system. The absolute voting requires that all the samples be in agreement. If one of the samples does not agree then a false Start bit is determined and the Rx Machine returns to the Start Bit search Mode. Once a Start bit is detected the Rx Machine will use the majority voting sampling window to receive the data bits.

Break Detection—If the input is low for the entire character frame including the stop Bit, then the Rx Machine will set Break Detected as well as Framing Error in the RECEIVE STATUS and LINE STATUS registers. It will push a NULL character onto the Rx FIFO with a framing-error and Break flag (As part of the Receive Flags). The Rx Machine then enters the Idle state. When it sees a mark it will set Break Terminated in RECEIVE STATUS and LINE STATUS registers and resume normal operation.

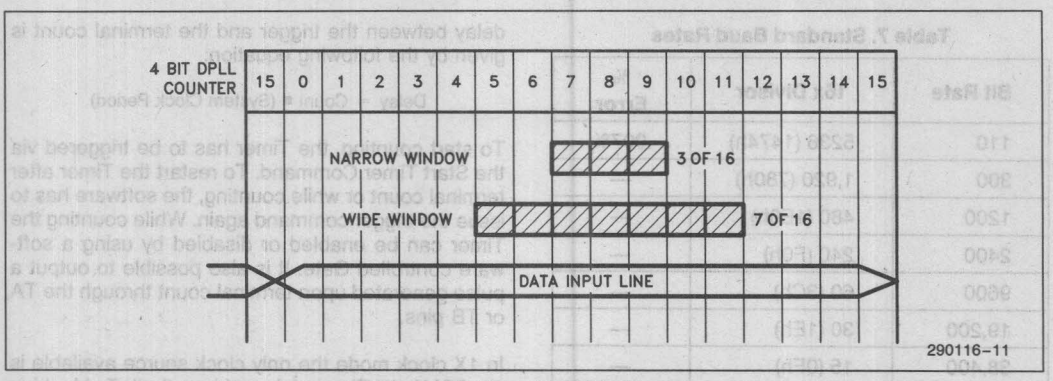


Figure 11. Sampling Windows

Control Characters—The Rx machine can generate a maskable interrupt upon reception of standard ASCII or EBCDIC control characters, or an Address marker is received in the uLAN mode. The Rx machine can also generate a maskable interrupt upon a match with programmed characters in the Address/Control Character 0 or Address/Control Character 1 registers.

Table 6. Control Character Recognition

CONTROL CHARACTER RECOGNITION	
A}	STANDARD SET ■ ASCII: 000X XXXX + 0111 1111 (ASCII DEL) (00 - 1FH + 7 FH) OR ■ EBCDIC: 00XX XXXX (00 - 3FH)
B}	User Programmed ■ ACR0, ACR1 XXXX XXXX REGISTERS

Baud-Rate Generators/Timers

The 82510 has two-on-chip, 16-bit baud-rate generators. Each BRG can also be configured as a Timer, and is completely independent of the other. This can be used when the Transmit and Receive baud rates are different. The mode, the output, and the source of each BRG is configurable, and can also be optionally output to external devices via the TA, TB pins (see Fig. 12. BRG Sources and Outputs).

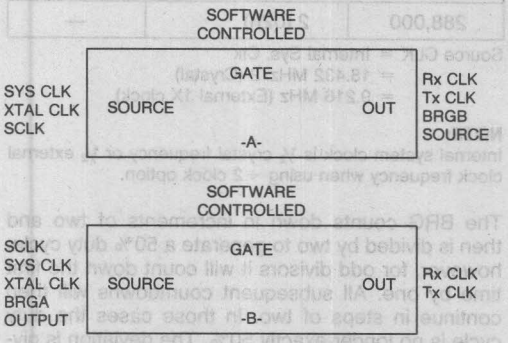


Figure 12. BRG Sources and Outputs

BAUD RATE GENERATION

The Baud Rate is generated by dividing the source clock with the divisor count. The count is loaded from the divisor count registers into a count down register. A 50% duty cycle is generated by counting down in steps of two. When the count is down to 2 the entire count is reloaded and the output clock is toggled. Optionally the two BRGs may be cascaded to provide a larger divisor. Note that this is the default configuration and used for 8250A/16450 emulation.

$$f_O = f_{in}/Divisor$$

where $f$in is the input clock frequency and Divisor is the count loaded into the appropriate count registers. System clock frequencies can be selected (4 → 9.216 MHz) to eliminate baud rate error for high baud rates.

Table 7. Standard Baud Rates

Bit Rate	16x Divisor	% Error
110	5236 (1474h)	.007%
300	1,920 (780h)	—
1200	480 (1E0h)	—
2400	240 (F0h)	—
9600	60 (3Ch)	—
19,200	30 (1Eh)	—
38,400	15 (0Fh)	—
56,000	10 (0Ah)	2.8%
288,000	2 (02h)	—

Source CLK = Internal Sys. Clk
= 18.432 MHz/2 (Crystal)
= 9.216 MHz (External 1X clock)

NOTE:
Internal system clock is ½ crystal frequency or ½ external clock frequency when using ÷2 clock option.

The BRG counts down in increments of two and then is divided by two to generate a 50% duty cycle; however, for odd divisors it will count down the first time by one. All subsequent countdowns will then continue in steps of two. In those cases the duty cycle is no longer exactly 50%. The deviation is given by the following equation:

$$\text{deviation} = 1/(2 \times \text{divisor})$$

The BRG can operate with any divisor between 1 and 65,535; however, for divisors between 1 and 3 the duty cycle is as follows:

Table 8. Duty Cycles

Divisor	Duty Cycle
3	33%
2	50%
1	Same as Source
0	FORBIDDEN

Timer Mode

Each of the 82510 BRGs can be used as Timers. The Timer is used to generate time delays by counting the internal system clock. When enabled the Timer uses the count from the Divisor/Count registers to count down to 1. Upon terminal count a maskable Timer Expired interrupt is generated. The delay between the trigger and the terminal count is given by the following equation:

$$\text{Delay} = \text{Count} \times (\text{System Clock Period})$$

To start counting, the Timer has to be triggered via the Start Timer Command. To restart the Timer after terminal count or while counting, the software has to issue the trigger command again. While counting the Timer can be enabled or disabled by using a software controlled Gate. It is also possible to output a pulse generated upon terminal count through the TA or TB pins.

In 1X clock mode the only clock source available is the SCLK pin. The serial machines (both Tx Machine and Rx Machine) can independently use one of two clock modes, either 1X or 16X. Also no configuration changes are allowed during operation as each write in the BRG configuration registers causes a reset signal to be sent to the BRG logic. The mode or source clocks may be changed only after a Hardware or Software reset. The Divisor (or count, depending upon the mode) may be updated during operation unless the particular BRG machine is being used as a clock source for one of the serial machines, and the particular serial machine is in operation at the time. Loading the count registers with "0" is forbidden in all cases, and loading it with a "1" is forbidden in the Timer Mode only.

SERIAL DIAGNOSTICS

The 82510 supports two modes of Loopback operation, Local Loopback and Remote Loopback as well as an Echo mode for diagnostics and improved throughput.

LOCAL LOOPBACK

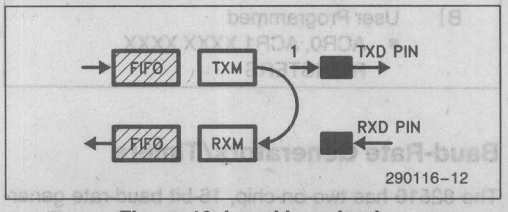


Figure 13. Local Loopback

The Tx Machine output and Rx Machine input are shorted internally, TXD pin output is held at Mark. This feature allows simulation of Transmission/Reception of characters and checks the Tx FIFO, Tx Machine, Rx Machine, and Rx FIFO along with the software without any external side effects. The modem outputs $\overline{OUT1}$, $\overline{OUT2}$, $\overline{DTR}$ and $\overline{RTS}$ are internally shorted to $\overline{RI}$, $\overline{DCD}$, $\overline{DSR}$ and $\overline{CTS}$ respectively. $\overline{OUT0}$ is held at a mark state.

REMOTE LOOPBACK

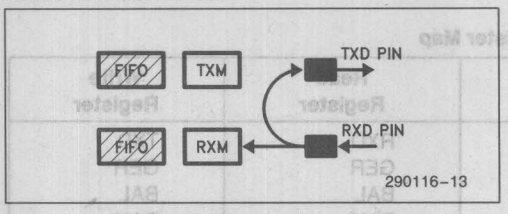


Figure 14. Remote Loopback

The TXD pin and RXD pin are shorted internally (the data is not sent on to the RX Machine). This feature allows the user to check the communications channel as well as the Tx and Rx pin circuits not checked in the Local Loopback mode.

AUTO ECHO

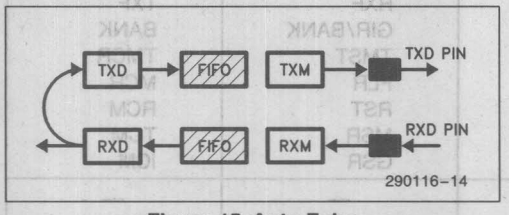


Figure 15. Auto Echo

In Echo Mode the received characters are automatically transmitted back. When the characters are read from the Rx FIFO they are automatically pushed back onto the Tx FIFO (the flags are also included). The Rx Machine baud rate must be equal to, or less than, the Tx Machine baud rate or some of the characters may be lost. The user has an option of preventing echo of special characters; Control Characters and characters with Errors.

Power Down Mode

The 82510 has a "power down" mode to reduce power consumption when the device is not in use. The 82510 powers down when the power down command is issued via the Internal Command Register (ICM). There are two modes of power down, Sleep and Idle.

In Sleep mode, even the system clock of the 82510 is shut down. The system clock source of the 82510 can either be the Crystal Oscillator or an external clock source. If the Crystal Oscillator is being used and the power down command is issued, then the 82510 will automatically enter the Sleep mode. If an external clock is being used, then the user must disable the external clock in addition to issuing the Power Down command, to enter the Sleep mode. The benefit of this mode is the increased savings in power consumption (typical power consumption in the Sleep mode is in the ranges of 100s of microAmps). However, upon wake up, the user must reprogram the device. To exit this mode the user can either issue a Hardware reset, or read the FIFO Level Register (FLR) and then issue a software reset. In either case the contents of the 82510 registers are not preserved and the device must be reprogrammed prior to operation. If the Crystal Oscillator is being used then the user must allow enough time for the oscillator to wake up before issuing the software reset.

The 82510 is in the idle mode when the Power Down command is issued and the system clock is still running (i. e. the system clock is generated externally and not disabled by the user). In this mode the contents of all registers and memory cells are preserved, however, the power consumption in this mode is greater than in the Sleep mode. Reading FLR will take the 82510 out of this mode.

NOTE:
The data read from FLR when exiting Power Down is invalid and should be ignored.

DETAILED REGISTER DESCRIPTION

Table 9. Register Map

Bank	Address	Read Register	Write Register
0 (NAS) 8250A/16450	0 (DLAB = 0)	RXD	TXD
	1 (DLAB = 0)	GER	GER
	0 (DLAB = 1)	BAL	BAL
	1 (DLAB = 1)	BAH	BAH
	2	GIR/BANK	BANK
	3	LCR	LCR
	4	MCR	MCR
	5	LSR	LSR
	6	MSR	MSR
	7	ACR0	ACR0
1 (WORK)	0	RXD	TXD
	1	RXF	TXF
	2	GIR/BANK	BANK
	3	TMST	TMCR
	4	FLR	MCR
	5	RST	RCM
	6	MSR	TCM
	7	GSR	ICM
2 (GENERAL CONF)	0	—	—
	1	FMD	FMD
	2	GIR/BANK	BANK
	3	TMD	TMD
	4	IMD	IMD
	5	ACR1	ACR1
	6	RIE	RIE
	7	RMD	RMD
3 (MODEM CONF)	0 (DLAB = 0)	CLCF	CLCF
	1 (DLAB = 0)	BACF	BACF
	0 (DLAB = 1)	BBL	BBL
	1 (DLAB = 1)	BBH	BBH
	2	GIR/BANK	BANK
	3	BBCF	BBCF
	4	PMD	PMD
	5	MIE	MIE
	6	TMIE	TMIE
	7	—	—

(1) ACRO is used in INS8250 as a Scratch-Pad Register
(2) DLAB = LCR Bit #7

The 82510 has thirty-five registers which are divided into four banks of register banks. Only one bank is accessible at any one time. The bank is selected through the BANK1-0 bits in the GIR/BANK register. The individual registers within a bank are selected by the three address lines (A2–0). The 82510 registers can be grouped into the following categories.

BANK ZERO 8250A/16450—COMPATIBLE BANK										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
TxD (33)	Tx Data bit 7	Tx Data bit 6	Tx Data bit 5	Tx Data bit 4	Tx Data bit 3	Tx Data bit 2	Tx Data bit 1	Tx Data bit 0	0	—
RxD (35)	Rx Data bit 7	Rx Data bit 6	Rx Data bit 5	Rx Data bit 4	Rx Data bit 3	Rx Data bit 2	Rx Data bit 1	Rx Data bit 0	0	—
BAL (11)	BRGA LSB Divide Count (DLAB = 1)								0	02H
BAH (12)	BRGA MSB Divide Count (DLAB = 1)								1	00H
GER (16)	0	0	Timer Interrupt Enable	Tx Machine Interrupt Enable	Modem Interrupt Enable	Rx Machine Interrupt Enable	Tx FIFO Interrupt Enable	Rx FIFO Interrupt Enable	1	00H
GIR/BANK (21)	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
LCR (2)	DLAB Divisor Latch Access bit	Set Break	Parity Mode bit 2	Parity Mode bit 1	Parity Mode bit 0	Stop bit Length bit 0	Character Length bit 1	Character Length bit 0	3	00H
MCR (32)	0	0	OUT 0 Complement	Loopback Control bit	OUT 2 Complement	OUT 1 Complement	RTS Complement	DTR Complement	4	00H
LSR (22)	0	TxM Idle	Tx FIFO Interrupt	Break Detected	Framing Error	Parity Error	Overrun Error	Rx FIFO Int Req	5	60H
MSR (27)	DCD Input Inverted	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State (H → L) Change in RI	State Change in DSR	State Change in CTS	6	00H
ACR0 (5)	Address or Control Character Zero								7	00H

BANK ONE—GENERAL WORK BANK										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
TxD (33)	Tx Data bit 7	Tx Data bit 6	Tx Data bit 5	Tx Data bit 4	Tx Data bit 3	Tx Data bit 2	Tx Data bit 1	Tx Data bit 0	0	—
RxD (35)	Rx Data bit 7	Rx Data bit 6	Rx Data bit 5	Rx Data bit 4	Rx Data bit 3	Rx Data bit 2	Rx Data bit 1	Rx Data bit 0	0	—
RxF (24)	—	Rx Char OK	Rx Char Noisy	Rx Char Parity Error	Address or Control Character	Break Flag	Rx Char Framing Error	Ninth Data bit of Rx Char	1	—
TxF (34)	Address Marker bit	Software Parity bit	Ninth bit of Data Char	0	0	0	0	0	1	—
GIR/BANK (21)	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
TMST (26)	—	—	Gate B State	Gate A State	—	—	Timer B Expired	Timer A Expired	3	30H
TMCR (31)	0	0	Trigger Gate B	Trigger Gate A	0	0	Start Timer B	Start Timer A	3	—
MCR (32)	0	0	OUT 0 Complement	Loopback Control bit	OUT 2 Complement	OUT 1 Complement	RTS Complement	DTR Complement	4	00H

NOTE:
The register number is provided as a reference number for the register description.

BANK ONE—GENERAL WORK BANK (Continued)

Register	7	6	5	4	3	2	1	0	Address	Default
FLR (25)	—	Rx FIFO Level			—	Tx FIFO Level			4	00H
RST (23)	Address/ Control Character Received	Address/ Control Character Match	Break Terminated	Break Detected	Framing Error	Parity Error	Overrun Error	Rx FIFO Interrupt Requested	5	00H
RCM (30)	Rx Enable	Rx Disable	Flush RxM	Flush Rx FIFO	Lock Rx FIFO	Open Rx FIFO	0	0	5	—
MSR (27)	DCD Complement	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State Change in RI	State Change in DSR	State Change in CTS	6	00H
TCM (29)	0	0	0	0	Flush Tx Machine	Flush Tx FIFO	Tx Enable	Tx Disable	6	—
GSR (20)	—	—	Timer Interrupt	TxM Interrupt	Modem Interrupt	RxM Interrupt	Tx FIFO Interrupt	Rx FIFO Interrupt	7	12H
ICM (28)	0	0	0	Software Reset	Manual Int Acknowledge Command	Status Clear	Power Down Mode	0	7	—

BANK TWO—GENERAL CONFIGURATION

Register	7	6	5	4	3	2	1	0	Address	Default
FMD (4)	0	0	Rx FIFO Threshold		0	0	Tx FIFO Threshold		1	00H
GIR/BANK (21)	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
TMD (3)	Error Echo Disable	Control Character Echo Disable	9-bit Character Length	Transmit Mode		Software Parity Mode	Stop Bit Length		3	00H
IMD (1)	0	0	0	0	Interrupt Acknowledge Mode	Rx FIFO Depth	ulan Mode Select	Loopback or Echo Mode of Operation	4	0CH
ACR1 (6)	Address or Control Character 1								5	00H
RIE (17)	Address/ Control Character Recognition Interrupt Enable	Address/ Control Character Match Interrupt Enable	Break Terminate Interrupt Enable	Break Detect Interrupt Enable	Framing Error Interrupt Enable	Parity Error Interrupt Enable	Overrun Error Interrupt Enable	0	6	1EH
RMD (7)	Address/Control Character Mode		Disable DPLL	Sampling Window Mode	Start bit Sampling Mode	0	0	0	7	00H

BANK THREE—MODEM CONFIGURATION

Register	7	6	5	4	3	2	1	0	Address	Default
CLCF (8)	Rx Clock Mode	Rx Clock Source	Tx Clock Mode	Tx Clock Source	0	0	0	0	0	00H
BACF (9)	0	BRGA Clock Source	0	0	0	BRGA Mode	0	0	1	04H
BBL (13)	BRGB LSB Divide Count (DLAB = 1)								0	05H
BBH (14)	BRGB MSB Divide Count (DLAB = 1)								1	00H

BANK THREE—MODEM CONFIGURATION (Continued)										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
GIR/BANK (21)	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
BBCF (10)	BRGB Clock Source		0	0	0	BRGB Mode	0	0	3	84H
PMD (15)	DCD/ICLK/ OUT 1 Direction	DCD/ICLK/ OUT 1 Function	DSR/TA/ OUT 0 Direction	DSR/TA/ OUT 0 Function	RI/SCLK Function	DTR/TB Function	0	0	4	FCH
MIE (19)	0	0	0	0	DCD State Change Int Enable	RI State Change Int Enable	DSR State Change Int Enable	CTS State Change Int Enable	5	0FH
TMIE (18)	0	0	0	0	0	0	Timer B Interrupt Enable	Timer A Interrupt Enable	6	00H

2

CONFIGURATION

These read/write registers are used to configure the device. They may be read at anytime; however, they may be written to only when the device is idle. Typically they are written to only once after system power up. They are set to default values upon Hardware or Software Reset (Default Wake-Up Mode). The default values are chosen so as to allow the 82510 to be fully software compatible with the IBM PC Async Adapter (INS 8250A/ 16450) when in the default wakeup mode. The 82510 can operate in the High Performance mode by programming the configuration registers as necessary.

The configuration options available to the user are listed below.

Table 11. Configuration Options

Interrupt Acknowledge Mode
- Automatic
- Manual

Receive
- Sampling Window Size
- Start Bit Detection Mode
- DPLL Disable/Enable

μLAN (8051)
Address Recognition
- Manual, Semi-Automatic, Automatic

Diagnostics
- Loopback
  - Remote
  - Local
- Echo
  - Yes/No
  - Disable Error Echo
  - Disable Control/Address Char. Echo

FIFO
- RX FIFO Depth
- RX, TX Threshold

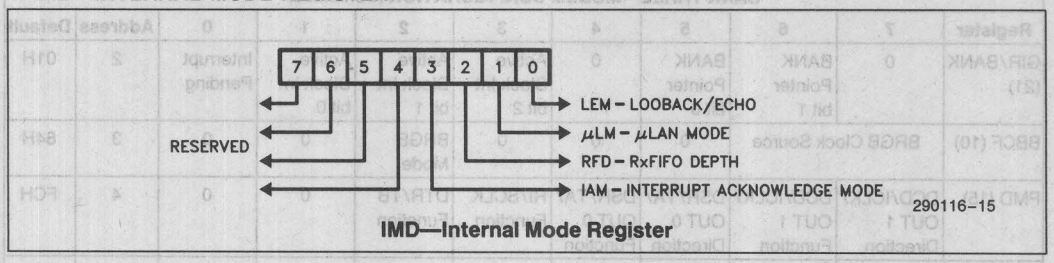
Clock Options
- RX, TX Clock Mode
  - 1X
  - 16X
- RX, TX Clock Source
  - BRGA
  - BRGB
- BRGA/B Operation Mode
  - Timer
  - BRG
- BRGA/B Divide Count
- BRGA/B Source
  - Sys Clock
  - SCLK Pin
  - BRGA Output (BRGB Only)

Control Character Recognition
- None
- Standard
  - ASCII
  - EBCDIC
- Two User Programmed

TX Operation
- RTS/CTS Control Manual, Semi-Automatic, Automatic
- Parity Mode
- Stop Bit Length
- Character Size

I/O Pins
- Select Function for Each Multifunction Pin
- Select Direction for Multifunction Pin (If Applicable)

1. IMD—INTERNAL MODE REGISTER



IMD—Internal Mode Register

This register defines the general device mode of operation. The bit functions are as follows:

7–4: Reserved

IAM: Interrupt Acknowledge Mode Bit

0 — Manual acknowledgement of pending interrupts

1 — Automatic acknowledgement of pending interrupts (upon CPU service)

This bit, when set, configures the 82510 for the automatic acknowledge mode. This causes the 82510 INT line to go low for two clock cycles upon service of the interrupt. After two clock cycles it is then updated. It is useful in the edge triggered mode. In manual acknowledgement mode the CPU must explicitly issue a command to clear the INT pin. (The INT pin then goes low for a minimum of two clock cycles until another enabled status register bit is set.)

RFD: Receive FIFO Depth

0 — Four Bytes
1 — One Byte

This bit configures the depth of the Rx FIFO. With a FIFO depth of one, the FIFO will act as a 1-byte buffer to emulate the 8250A.

ULM: uLAN Mode

0 — Normal Mode
1 — uLAN Mode

This bit, enables the 82510 to recognize and/or match address using the 9-bit MCS-51 asynchronous protocol.

LEM: Loopback/Echo Mode Select

This bit selects the mode of loopback operation, or the mode of echo operation; depending upon which operation mode is selected by the Modem Control register bit LC.

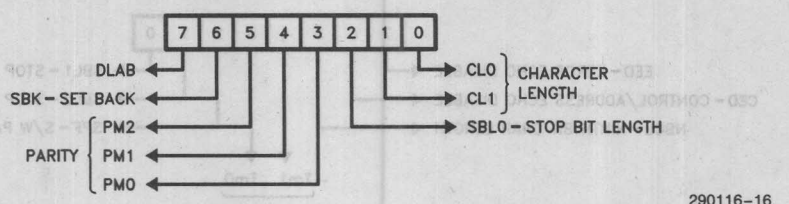
In *loopback* mode (Modem Control register bit LC = 1) it selects between local and remote loopback.

0 — Local Loopback
1 — Remote Loopback

In *echo* mode (Modem Control register bit LC = 0) it selects between echo or non-echo operation.

0 — No Echo
1 — Echo Operation

2. LCR—LINE CONFIGURE REGISTER



LCR—Line Configure Register

This register defines the basic configuration of the serial link.

DLAB—Divisor Latch Access Bit—This bit, when set, allows access to the Divisor Count registers BAL,BAH;BBL,BBH registers.

SBK—Set Break Bit—This bit will force the TxD pin low. The TxD pin will remain low (regardless of all activities) until this bit is reset.

PM2—PM0—Parity Mode Bits—These three bits combine with the SPF bit of the Transmit Mode register to define the various parity modes. See Table 12.

Table 12. Parity Modes

PM0	SPF	PM2	PM1	Function
0	X	X	X	No Parity
1	0	0	0	Odd Parity
1	0	0	1	Even Parity
1	0	1	0	High Parity
1	0	1	1	Low Parity
1	1	0	0	Software Parity

SBL0—Stop Bit Length—This bit, together with SBL1 and SBL2 bits of the Transmit Mode register, defines the Stop-bit lengths for transmission. The Rx machine can identify 3/4 stop bit or more. See Table 13.

Table 13. Stop Bit Length

SBL2	SBL1	SBL0	Stop Bit Length 16X	1X
0	0	0	4/4	—
0	0	1	6/4 or 8/4*	—
0	1	0	3/4	1
0	1	1	4/4	1
1	0	0	5/4	1
1	0	1	6/4	1
1	1	0	7/4	1
1	1	1	8/4	2

*6/4 if character length is 5 bits; else 8/4

CL0—CL1—Character Length Bits—These bits, together with the Transmit Mode register bit NBCL, define the character length. See Table 14.

Table 14. Character Length

NBCL	CL1	CL0	Character Length
0	0	0	5 BITS
0	0	1	6 BITS
0	1	0	7 BITS
0	1	1	8 BITS
1	0	0	9 BITS

This register together with the Line Configure Register defines the Tx machine mode of operation.

EED—Error Echo Disable—Disables Echo of characters received with errors (valid in echo mode only).

CED—Control Character Echo Disable—Disables Echo of characters recognized as control characters (or address characters in uLAN mode). Valid in echo mode only.

NBCL—Nine-Bit Length—This bit, coupled with LCR (CL0, CL1), selects Transmit/Receive character length of nine bits. See Table 14.

TM1—TM0—Transmit Mode—These bits select one of three modes of control over the $\overline{CTS}$ and $\overline{RTS}$ lines.

00—Manual Mode—In this mode the CPU has full control of the Transmit operation. The CPU has to explicitly enable/disable transmission, and activate/check the $\overline{RTS}$/$\overline{CTS}$ pins.

01—Reserved

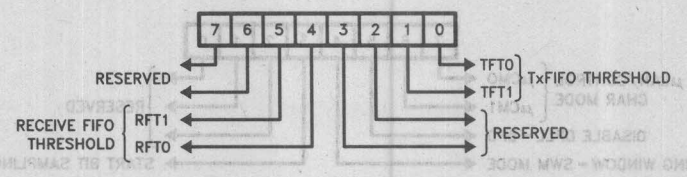
10—Semi-Automatic Mode—In this mode the 82510 transmits only when $\overline{CTS}$ input is active. The 82510 activates the $\overline{RTS}$ output as long as transmission is enabled.

11—Automatic Mode—In this mode the 82510 transmits only when $\overline{CTS}$ input is active. The $\overline{RTS}$ output is activated only when transmission is enabled and there is more data to transmit.

SPF—Software Parity Force—This bit defines the parity modes along with the PM0, PM1, and PM2 bits of the LCR register. When software parity is enabled the software must determine the parity bit through the TxF register on transmission, or check the parity bit in RxF upon reception. See Table 12.

SBL2—SBL1—Stop Bit Length—These bits, together with the SBL0 bit of the LCR register define the stop bit length. See Table 13.

4. FMD—FIFO MODE REGISTER



FMD—FIFO Mode Register

This register configures the Tx and Rx FIFO's threshold levels—the occupancy levels that can cause an interrupt.

7—6—Reserved

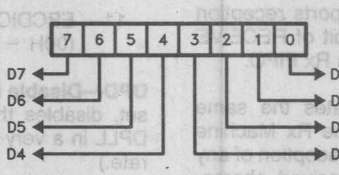
RFT1—RFT0—Receive FIFO Threshold—When the Rx FIFO occupancy is greater than the level indicated by these bits the Rx FIFO Interrupt is activated.

3—2—Reserved

TFT1—TFT0—Transmit FIFO Threshold—When the TX FIFO occupancy is less than or equal to the level indicated by these bits the Tx FIFO Interrupt is activated.

2

5. ACR0—ADDRESS/CONTROL CHARACTER REGISTER 0



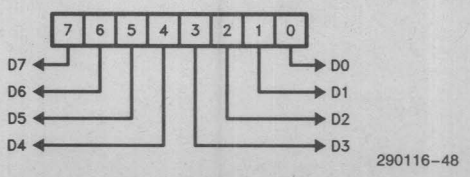
ARC0—Address/Control Character Register 0

This register contains a byte which is compared to each received character. The exact function depends on the configuration of the IMD register.

In normal mode this register may be used to program a special control character; a matched character will be reported in the RECEIVE STATUS register. The maximum length of the control characters is eight bits. If the length is less than eight bits then the character must be right justified and the leading bits be filled with zeros.

In uLAN mode this register contains the eight-bit station address for recognition. In this mode only incoming address characters (i.e., characters with address bit set) will be compared to these register. The PCRF bit in the RECEIVE STATUS register will be set when an Address or Control Character match occurs.

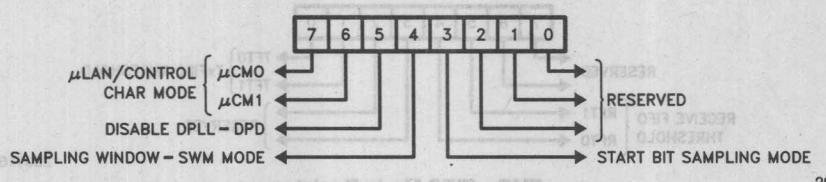
6. ACR1—ADDRESS/CONTROL CHARACTER REGISTER 1



ARC1—Address/Control Character Register 1

NOTE:
This register is identical in function to ACR0.

7. RMD—RECEIVE MACHINE MODE REGISTER



RMD—Receive Machine Mode Register

This register defines the Rx Machine mode of operation.

uCM0, uCM1—uLAN/Control Character Recognition Mode—In normal mode it defines the Control Character recognition mode. In ulan mode they define modes of address recognition.

In *uLAN* mode: selects the mode of address recognition.

00—Manual Mode—Rx Machine reports reception of any address character, via CRF bit of RECEIVE STATUS register, and writes it to the Rx FIFO.

01—Semi-Automatic Mode—Operates the same as Manual Mode but, in addition, the Rx Machine OPENS (unlocks) the Rx FIFO upon reception of any address characters. Subsequent received characters will be written into the FIFO. (Note: it is the user's responsibility to LOCK the FIFO if the address character does not match the station's address.)

10—Automatic Mode—The Rx Machine will OPEN (unlock) the Rx FIFO upon Address Match. In addition the Rx Machine LOCKs the Rx FIFO upon recognition of address mismatch; i.e., it controls the flow of characters into the Rx FIFO depending upon the results of the address comparison. If a match occurs it will allow characters to be sent to the FIFO; if a mismatch occurs it will keep the characters out of the FIFO by LOCKING it.

11—Reserved

In *normal* Mode: selects the mode of Standard Set Control Character Recognition (programmed control characters are always recognized).

00— No Standard Set Control Characters Recognized.

01— ASCII Control Characters (00H—1 FH + 7FH).

10— Reserved.

11— EBCDIC Control Character Recognized (00H – 3FH).

DPD—Disable Digital Phase Locked Loop—When set, disables the DPLL machine. (Note: using the DPLL in a very noisy media, may increase the error rate.)

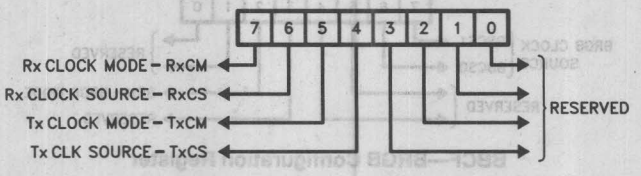
SWM—Sampling Window Mode—This bit controls the mode of data sampling:

0—Small Window, 3/16 sampling.
1—Large Window, 7/16 sampling.

SSM—Start Bit Sampling Mode—This bit controls the mode of Start Bit sampling.

0— Majority Voting for start bit. In this mode a majority of the samples determines the bit.
1— In this mode if one of the bit samples is not '0', the start bit will not be detected.

8. CLCF—CLOCKS CONFIGURE REGISTER



CLCF—Clocks Configure Register

This register defines the Transmit and Receive Code modes and sources.

RxCM—Rx Clock Mode—This bit selects the mode of the receive clock which is used to sample the received data.

0— 16X Mode.
1— 1X Mode. In this mode the receive data must be synchronous to the Rx Clock; supplied via the SCLK pin.

RxCS—Rx Clock Source—This bit selects the source of the internal receive clock in the case of 16X mode (as programmed by the RxCM bit above).

0—BRGB Output
1—BRGA Output

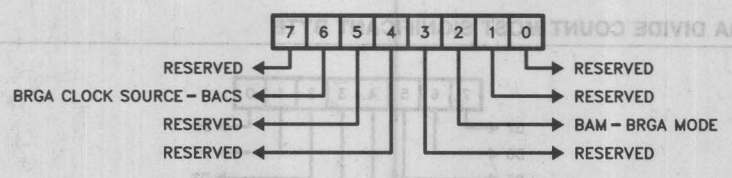
TxCM—Transmit Clock Mode—This bit selects the mode of the Transmit Data Clock, which is used to clock out the Transmit Data.

0— 16X Mode
1— 1X Mode. In this mode the Transmit data is synchronous to the Serial Clock; supplied via the SCLK pin.

TxCS—Transmit Clock Source—Selects the source of internal Transmit Clock in case of 16X mode.

0—BRGB Output.
1—BRGA Output.

9. BACF—BRGA CONFIGURATION REGISTER



BACF—BRGA Configuration Register

This register defines the BRGA clock sources and the mode of operation.

BACS—BRGA Clock Source—Selects the input clock source for Baud Rate Generator A.

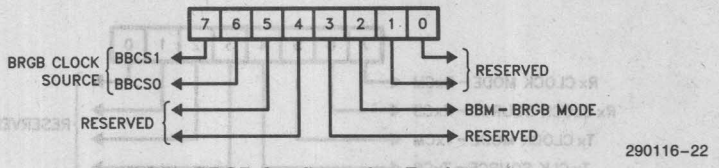
0—System Clock
1—SCLK Pin

This bit has no effect if BRGA is configured as a timer.

BAM—BRGA Mode of Operation—Selects between the Timer mode or the Baud Rate Generator Mode.

0— Timer Mode (in this mode the input clock source is always the system clock).
1— Baud Rate Generator Mode

10. BBCF—BRGB CONFIGURATION REGISTER



BBCF—BRGB Configuration Register

This register defines the BRGB clock sources and mode of operation. (Note: BRGB can also take its Input Clock from the output of BRGA.)

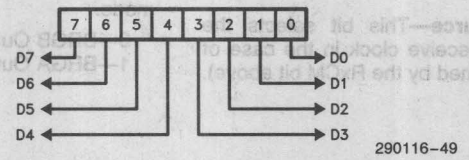
BBCS1, BBCS0—These two bits together define the input Clock Sources for BRGB. These bits have no effect when in the timer mode.

00— System Clock
01— SCLK Pin
10— BRGA Output
11— Reserved

BBM—BRGB Mode of Operation.

0— Timer Mode (In this mode the input clock source is always the system clock.)
1— BRG Mode

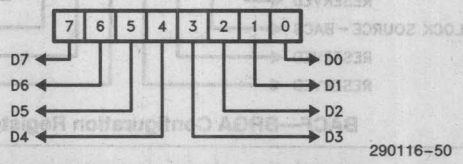
11. BAL—BRGA DIVIDE COUNT LEAST SIGNIFICANT BYTE



BAL—BRGA Divide Count Low Byte

This register contains the least significant byte of the BRGA divisor/count.

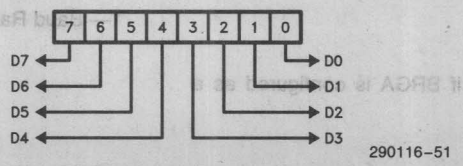
12. BAH—BRGA DIVIDE COUNT MOST SIGNIFICANT BYTE



BAH—BRGA Divide Count High Byte

This register contains the most significant byte of the BRGA divisor/count.

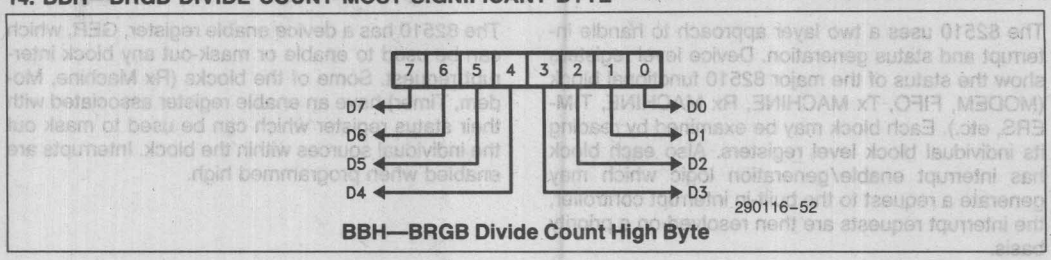
13. BBL—BRGB DIVIDE COUNT LEAST SIGNIFICANT BYTE



BBL—BRGB Divide Count Low Byte

This register contains the least significant byte of the BRGB divisor/count.

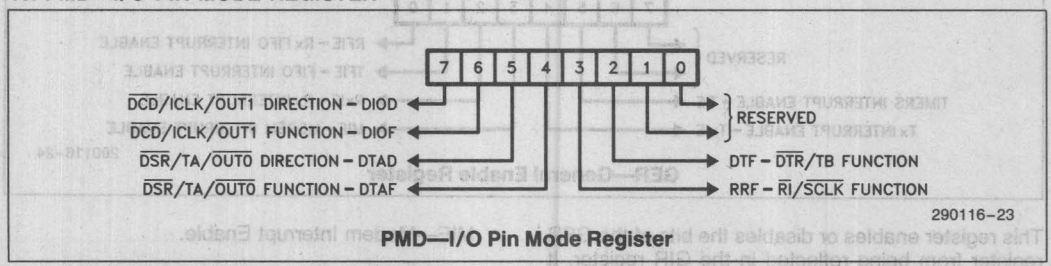
14. BBH—BRGB DIVIDE COUNT MOST SIGNIFICANT BYTE



BBH—BRGB Divide Count High Byte

This register contains the most significant byte of the BRGB divisor/count.

15. PMD—I/O PIN MODE REGISTER



PMD—I/O Pin Mode Register

This register is used to configure the direction and function of the multifunction pins. The following options are available on each pin.

1. Direction: Input or Output Pin.
   - 0— Defines the Pin as an output pin (general purpose or special function).
   - 1— Defines the pin as an input pin.
2. Function: General purpose or special purpose pin (no effect if the pin is programmed as an Input).
   - 0— special function output pin.
   - 1— general purpose output pin.
     DIOD—DCD/ICLK/OUT1 Direction.
   - 0— Output: ICLK or OUT1 (depending on bit DIOF)
   - 1— Input: DCD.
     DIOF—DCD/ICLK/OUT1 Function (output mode only).
   - 0— ICLK (Output of the Internal System Clock).
   - 1— OUT1 general purpose output, Controlled by MODEM CONTROL Register
     DTAD—DSR/TA/OUT0 Direction.
   - 0— Output: TA or OUT0 (Dependent upon DTAF).
   - 1— Input: DSR.
     DTAF—DSR/TA/OUT0 Direction (output mode only).
   - 0— TA (BRGA Output or Timer A Termination Pulse).
   - 1— OUT0 (general purpose output, controlled by MODEM CONTROL).
     RRF—RI/SCLK Function
   - 0— SCLK (Receive and/or Transmit Clock)
   - 1— RI
     DTF—DTR/TB Function
   - 0— TB (BRGB Output Clock on Timer B termination pulse depending upon the mode of BRGB).
   - 1— DTR

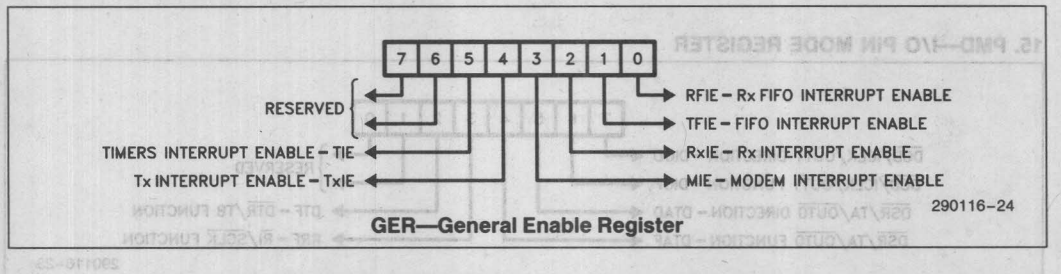
INTERRUPT/STATUS REGISTERS

The 82510 uses a two layer approach to handle interrupt and status generation. Device level registers show the status of the major 82510 functional block (MODEM, FIFO, Tx MACHINE, Rx MACHINE, TIMERS, etc.). Each block may be examined by reading its individual block level registers. Also each block has interrupt enable/generation logic which may generate a request to the built-in interrupt controller, the interrupt requests are then resolved on a priority basis.

Interrupt Masking

The 82510 has a device enable register, GER, which can be used to enable or mask-out any block interrupt request. Some of the blocks (Rx Machine, Modem, Timer) have an enable register associated with their status register which can be used to mask out the individual sources within the block. Interrupts are enabled when programmed high.

16. GER—GENERAL ENABLE REGISTER



GER—General Enable Register

This register enables or disables the bits of the GSR register from being reflected in the GIR register. It serves as the device enable register and is used to mask the interrupt requests from any of the 82510 block (See Figure 1).

TIE—Timers Interrupt Enable

TxIE—Transmit Machine Interrupt Enable.

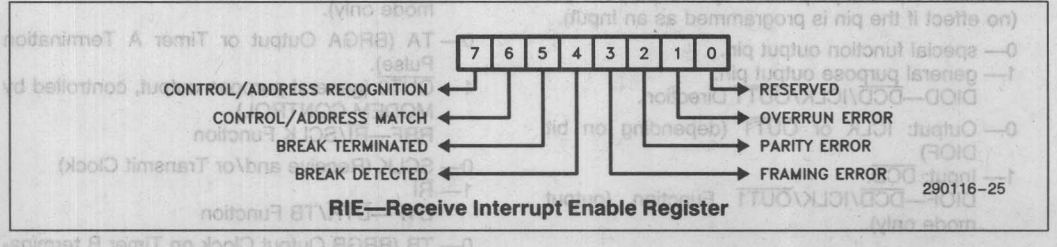
MIE—Modem Interrupt Enable.

RxIE—Rx Machine Interrupt Enable.

TFIE—Transmit FIFO Interrupt Enable.

RFIE—Receive FIFO Interrupt Enable.

17. RIE—RECEIVE INTERRUPT ENABLE REGISTER



RIE—Receive Interrupt Enable Register

This register enables interrupts from the Rx Machine. It is used to mask out interrupt requests generated by the status bits of the RST register.

CRE—Control/uLAN Address Character Recognition Interrupt Enable.—Enables Interrupt when CRF bit of RST register is set.

PCRE—Programmable Control/Address Character Match Interrupt Enable.—Enables Interrupt on PCRF bit of RST.

BkTe—Break Termination Interrupt Enable.

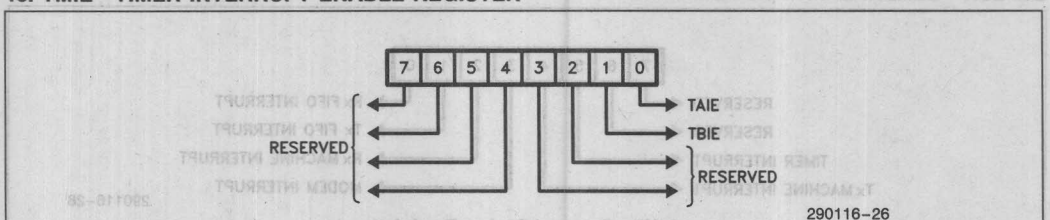
BkDE—Break Detection Interrupt Enable—Enable Interrupt on BkD bit of RST.

FEE—Framing Error Enable—Enable Interrupt on FE bit of RST.

PEE—Parity Error Enable—Enable Interrupt on PE bit of RST.

OEE—Overrun Error Enable—Enable Interrupt on OE bit of RST.

18. TMIE—TIMER INTERRUPT ENABLE REGISTER



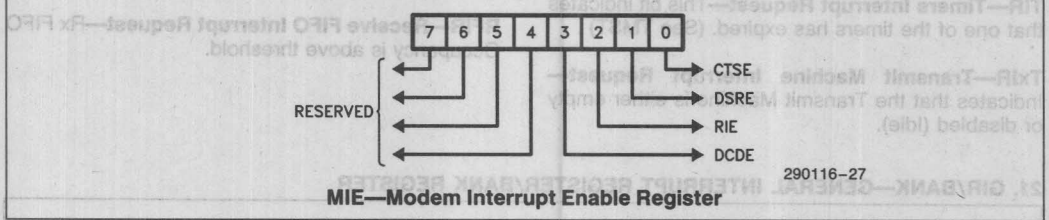
TMIE—Timers/Interrupt Enable Register

This is the enable register for the Timer Block. It is used to mask out interrupt requests generated by the status bits of the TMST register.

TBIE—Timer B Expired Interrupt Enable—Enables Interrupt on TBEx bit of TMST.

TAIE—Timer A Expired Interrupt Enable—Enables Interrupt on TAEx bit of TMST.

19. MIE—MODEM INTERRUPT ENABLE REGISTER



MIE—Modem Interrupt Enable Register

This register enables interrupts from the Modem Block. It is used to mask out interrupt requests generated by the status bits of the MODEM STATUS register.

DCDE—Delta $\overline{\text{DCD}}$ Interrupt Enable—Enables Interrupt on DDCD bit of MODEM STATUS.

RIE—Delta $\overline{\text{RI}}$ Interrupt Enable—Enables Interrupt on DRI bit of MODEM STATUS.

DSRE—Delta $\overline{\text{DSR}}$ Interrupt Enable—Enables Interrupt on $\overline{\text{DSR}}$ bit of MODEM STATUS.

CTSE—Delta $\overline{\text{CTS}}$ Interrupt Enable—Enables Interrupt on DCTS bits of MODEM STATUS.

STATUS/INTERRUPT

The 82510 has two device status registers, which reflect the overall status of the device, and five block status registers. The two device status registers, GSR and GIR, and supplementary in function. GSR reflects the interrupt status of all blocks, whereas GIR depicts the highest priority interrupt only. GIR is updated after the GSR register; the delay is of approximately two clock cycles.

GER masks-out interrupts from GIR; it does not have any effect on the bits in GSR. The only way that the bits can be masked out in GSR (i.e., not appear in GSR) is if they are masked out at the lower level.

TIR—Timers Interrupt Request—This bit indicates that one of the timers has expired. (See TMST)

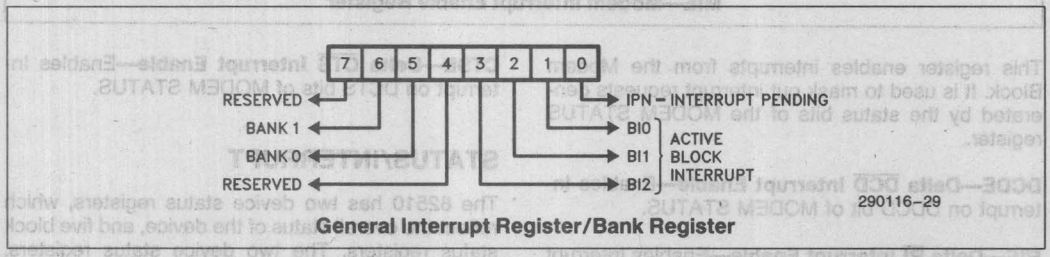
TxIR—Transmit Machine Interrupt Request—Indicates that the Transmit Machine is either empty or disabled (Idle).

RxIR—Receive Machine Interrupt Request—(As reflected in RST.)

TFIR—Transmit FIFO Interrupt Request—Tx FIFO occupancy is below or equal to threshold.

RFIR—Receive FIFO Interrupt Request—Rx FIFO Occupancy is above threshold.

21. GIR/BANK—GENERAL INTERRUPT REGISTER/BANK REGISTER



General Interrupt Register/Bank Register

This register holds the highest priority enabled pending interrupt from GSR. In addition it holds a pointer to the current register segment. Writing into this register will update only the BANK bits.

BANK1, BANK0—Bank Pointer Bits—These two bits point to the currently accessible register bank. The user can read and write to these bits. The address of this register is always two within all four register banks.

BI2, BI1, BI0,—Interrupt Bits 0–2—These three bits reflect the highest priority enabled pending interrupt from GSR.

101: Timer Interrupt (highest priority)
100: Tx Machine Interrupt
011: Rx Machine Interrupt
010: Rx FIFO Interrupt
001: Tx FIFO Interrupt
000: Modem Interrupt (lowest priority)

IPN—Interrupt Pending—This bit is active low. It indicates that there is an interrupt pending. The interrupt logic asserts the INT pin as soon as this bit goes active. (Note: the GIR register is continuously updated; so that, while the user is serving one interrupt source, a new interrupt with higher priority may enter GIR and replace the older interrupt.)

RST register (BITS 1–7) and LSR register (BITS 1–4) are cleared. This register is provided for compatibility with the INS8250A.

TxSt—Transmit Machine Status Bit—Same as TxIR bit of GSR register. If high it indicates that the Transmit Machine is in Idle State. (Note: Idle may indicate that the TxM is either empty or disabled.

TFSt—Transmit FIFO Status—Same as TFIR bit of GSR. It indicates that the Transmit FIFO level is equal to or below the Transmit FIFO Threshold. There are two ways to disable the transmit FIFO status from being reflected in GIR:

1. Writing a "0" to the TFIE bit of the GER register
2. Dynamically by using the Tx FIFO HOLD INTERRUPT logic. When the Tx FIFO is in the Hold State, no interrupts are generated regardless of the TFIR and TFIE bits.

The Transmit FIFO enters the Hold State when the CPU reads the GIR register and the source of the interrupt is Tx FIFO. To Exit, the CPU must drop the

BkD—Break Detected—See BkD bit in RST register for full explanation. The BkD bit in RST register is the same as this bit.

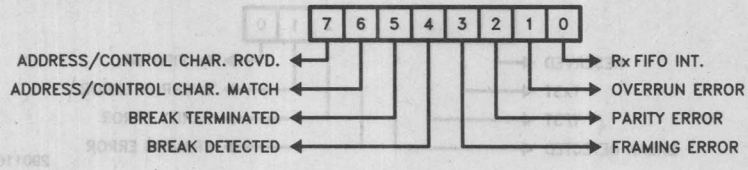
FE—Framing Error Detected—See FE bit in RST register for a full explanation. The FE bit in RST register is the same as this bit.

PE—Parity Error Detected—See PE bit in RST register for full explanation. The PE bit in RST register is the same as this bit.

OE—Overrun Error—See OE bit in RST register for full explanation. The OE bit in RST register is the same as this bit.

RFIR—Receive FIFO Interrupt Request—This bit is identical to RFIR bit of GSR. It indicates that the RX FIFO level is above the Rx FIFO Threshold. This bit is forced LOW during any READ from the Rx FIFO. A zero written to this bit will acknowledge an Rx FIFO interrupt.

23. RST—RECEIVE MACHINE STATUS REGISTER



RST—Receive Machine Status Register

This register displays the status of the Receive Machine. It reports events that have occurred since the RST was cleared. This register is cleared when it is read except for BIT0, Rx FIFO interrupt. Each bit in this register, when set, can cause an interrupt. Five bits of this register are shared with the LSR register.

CRF—Control/Address Character Received— When enabled, this bit can cause an interrupt if a control character or address character is received.

In uLAN Mode: indicates that an address character has been received.
In normal Mode: indicates that a standard control character (either ASCII or EBCDIC) has been received.

PCRF—Programmed Control/Address Character Received—This bit, when enabled, will cause an interrupt when an address or control character match occurs.

In uLAN Mode: indicates that an address character equal to one of the registers ACR0 or ACR1 has been received.
In normal Mode: indicates that a character which matches the registers ACR0 or ACR1 has been received.

BkT—Break Terminated—This bit indicates that a break condition has been terminated.

BkD—Break Detected—This bit indicates that a Break Condition has been detected, i.e., RxD input was held low for one character frame plus a stop BIT.

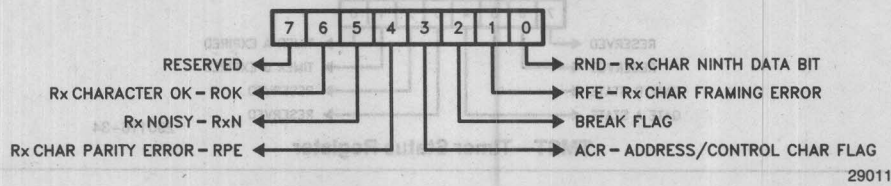
FE—Framing Error—This bit indicates that a received character did not have a valid stop bit.

PE—Parity Error—Indicates that a received character had a parity error.

OE—Overrun Error—Indicates that a received character was lost because the Rx FIFO was full.

RFIR—Receive FIFO Interrupt Request—Same as the RFIR bit of LSR register.

24. RXF—RECEIVED CHARACTER FLAGS



RxF—Receive Flags Register

This register contains additional information about the character in the RXD register. It is loaded by the Rx Machine simultaneously with the RXD register.

ROK—Received Character OK—This bit indicates that the character in RXD no parity or framing error. The parity error is not included in the s/w parity mode.

RxN—Received Character Noisy—The character in RXD was noisy. This bit, valid only in 16X sampling mode, indicates that the received character had non-identical samples for at least one of its bits.

RPE—Receive Character Parity Error—This bit indicates that the RxD character had a Parity Error. However, in S/W Parity mode it holds the received parity bit as is.

ACR—Address/Control Character Marker—This bit indicates that the character in RXD is either:

A control Character—in normal Mode.
An Address Character in uLAN Mode.

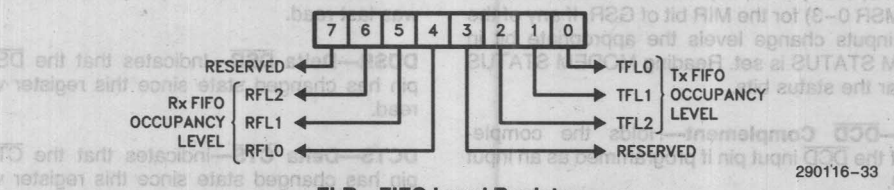
RFE—Receive Character Framing Error—Indicates that no Stop bit was found for the character in RXD.

NOTE:
A Framing Error will be generated for the first character of the Break sequence.

RND—Ninth Bit of Received Character—The most significant bit of the character in RXD is written into this bit. This bit is zero for characters with less than nine bits.

BKF—Break Flag—Indicates that the character is part of a "break" sequence.

25. FLR—FIFO LEVEL REGISTER



FLR—FIFO Level Register

This register holds the current Receive and Transmit FIFO occupancy levels.

RFL2, RFL1, RFL0—Receive FIFO Level of Occupancy—These three bits indicate the level of Occupancy of the Rx FIFO. The valid range is zero (000) to four (100).

TFL2, TFL1, TFL0—Transmit FIFO Level of Occupancy—These three bits indicate the level of occupancy in the transmit FIFO. The valid range is zero (000) to four (100).

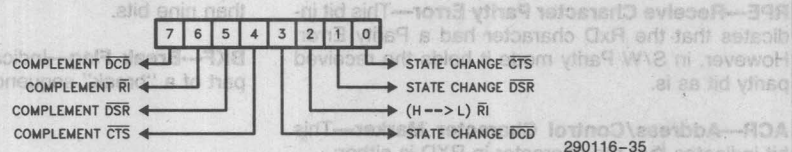
GBS—Gate B State—This bit does not generate an interrupt. It indicates the counting state of the software gate of Timer B, as written through the TMCR register.

0—counting disabled
1—counting enabled

TBEx—Timer B Expired—When Set generates an interrupt through TIR bit of GSR. Indicates that Timer B count has expired. This bit is set via the terminal count pulse generated by the timer when it terminates counting.

TAEx—Timer A Expired—Same as TBEx except it refers to Timer A.

27. MSR—MODEM/I/O PINS REGISTER



MSR—Modem/I/O Pins Status Register

This register holds the status of the Modem input pins (CTS, DCD, DSR, RI). It is the source of interrupts (MSR 0–3) for the MIR bit of GSR. If any of the above inputs change levels the appropriate bit in MODEM STATUS is set. Reading MODEM STATUS will clear the status bits.

DCDC—$\overline{DCD}$ Complement—Holds the complement of the $\overline{DCD}$ input pin if programmed as an input in PMD.

DRIC—Holds the complement of the $\overline{RI}$ input pin if programmed as an input in PMD.

DSRC—$\overline{DSR}$ Complement—Holds the complement of the $\overline{DSR}$ input pin if configured as an input in PMD.

CTSC—$\overline{CTS}$ Complement—Holds the complement of the $\overline{CTS}$ pin.

DDCD—Delta $\overline{DCD}$—Indicates that the $\overline{DCD}$ input pin has changed state since this register was last read.

DRI—Delta $\overline{RI}$—Indicates that there was a high-to-low transition on the $\overline{RI}$ input pin since the register was last read.

DDSR—Delta $\overline{DSR}$—Indicates that the $\overline{DSR}$ input pin has changed state since this register was last read.

DCTS—Delta $\overline{CTS}$—Indicates that the $\overline{CTS}$ input pin has changed state since this register was last read.

COMMAND REGISTERS

The command registers are write only; they are used to trigger an operation by the device. Once the operation is started the register is automatically reset. There is a device level register as well as four block command registers. It is recommended that only one command be issued during a write cycle.

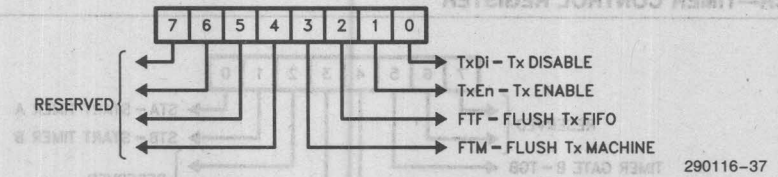
device reset; the effect is identical to the hardware reset (except for strapping options). The reset lasts four clocks and puts the device into the Default Wake-up Mode.

INTA—Interrupt Acknowledge—This command is an explicit acknowledgement of the 82510's interrupt request. It forces the INT pin inactive for two

StC—Status Clear—Clears the following status registers: RST, MSR, and TMST.

PDM—Power Down—This command forces the device into the power-down mode. Refer to the functional description for details.

29. TCM—TRANSMIT COMMAND REGISTER



TCM—Transmit Command Register

This register controls the operation of the Transmit Machine.

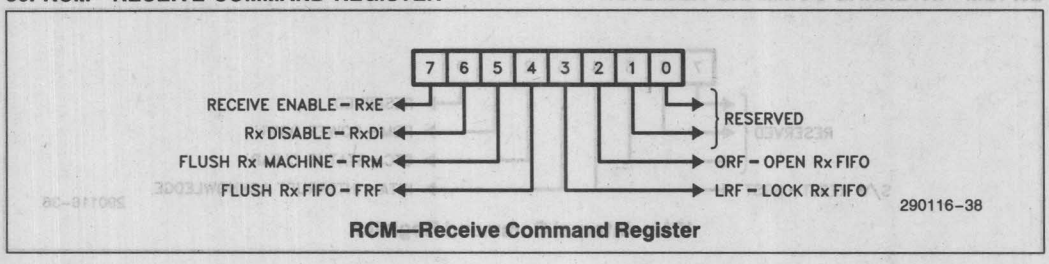
FTM—Flush Transmit Machine—Resets the Transmit Machine logic (but not the registers or FIFO) and enables transmission.

FTF—Flush Transmit FIFO—Clears the Tx FIFO.

TxEN—Transmit Enable—Enables Transmission by the Transmit Machine.

TxDi—Transmit Disable—Disables transmission. If transmission is occurring when this command is issued the Tx Machine will complete transmission of the current character before disabling transmission.

30. RCM—RECEIVE COMMAND REGISTER



RCM—Receive Command Register

This register controls the operation of the Rx machine.

RxE—Receive Enable—Enables the reception of characters.

RxDi—Receive Disable—Disables reception of data on RXD pin.

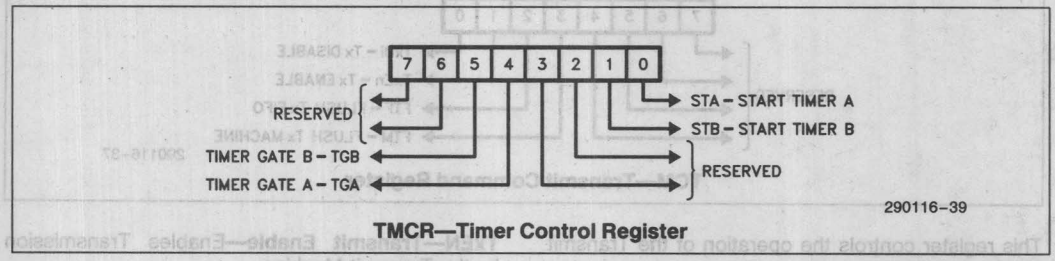
FRM—Flush Receive Machine—Resets the Rx Machine logic (but not registers and FIFOs), enables reception, and unlocks the receive FIFO.

FRF—Flush Receive FIFO—Clears the Rx FIFO.

LRF—Locks Rx FIFO—Disables the write mechanism of the Rx FIFO so that characters subsequently received are not written to the Rx FIFO but are lost. However, reception is not disabled and complete status/event reporting continues. (This command may be used in the uLAN mode to disable loading of characters into the Rx FIFO until an address match is detected.)

ORF—Open (Unlock) Rx FIFO—This command enables or unlocks the write mechanism of the Rx FIFO.

31. TMCR—TIMER CONTROL REGISTER



TMCR—Timer Control Register

This register controls the operation of the two 82510 timers. It has no effect when the timers are configured as baud—rate generators. TGA and TGB are not reset after command execution.

TGB—Timer-B Gate—This bit serves as a gate for Timer B operation:

1—enables counting
0—disables counting

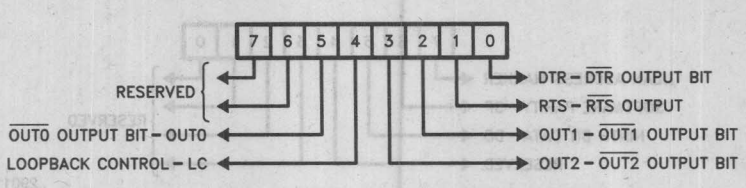
TGA—Timer-A Gate—This bit serves as a gate for Timer-A operation:

1—enables counting
0—disables counting

STB—Start Timer B—This command triggers timer B. At terminal count a status bit is set in TMST (TBEx).

STA—Start Timer A—This command triggers timer A. At terminal count a status bit is set in TMST (TAEx).

32. MCR—MODEM CONTROL REGISTER



MCR—Modem Control Register

This register controls the modem output pins. With multi—function pins it affects only the pins configured as general purpose output pins. All the output pins invert the data, i.e. their output will be the complement of the data written into this register.

OUT0—OUT0 Output Bit—This bit controls the OUT0 pin. The output signal is the complement of this bit.

LCB Loopback Control Bit—This bit puts the 82510 into loopback mode. The particular type of loopback is selected via the IMD register.

OUT2—OUT2 Output Bit—This bit controls the OUT2 pin. The output signal is the complement of this bit.

OUT1—OUT1 Output Bit—This bit controls the OUT1 pin. The output signal is the complement of this bit.

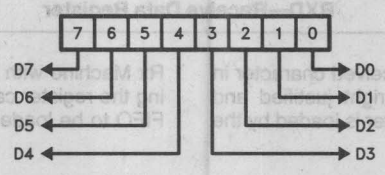
RTS—RTS Output Bit—This bit controls the RTS pin. The output signal is the complement of this bit.

DTR—DTR Output Bit—This bit controls the DTR pin. The output signal is the complement of this bit.

DATA REGISTERS

The data registers hold data or other information and may be accessed at any time.

33. TXD—TRANSMIT DATA REGISTER



TXD—Transmit Data Register

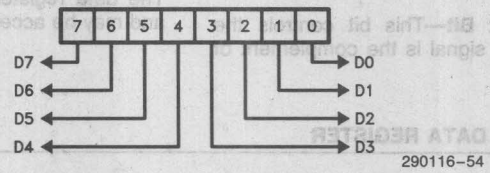
This register holds the next data byte to be pushed into the Transmit FIFO. For character formats with more than eight bits of data, or with additional components (S/W Parity, Address Marker Bit) the additional data bits should be written into the TxF register. When a byte is written to this register its contents, along with the contents of the TxF register, are pushed to the top of the Transmit FIFO. This register is write only.

TxD register is written to by the CPU.

uLAN—uLAN Address Marker Bit—This bit is transmitted in uLAN mode as the address marker bit.

D8—Ninth Bit of Data—In nine-bit character length mode this bit is transmitted as the MSB (D8) bit.

35. RXD—RECEIVE DATA REGISTER



RXD—Receive Data Register

This register holds the earliest received character in the Rx FIFO. The character is right justified and leading bits are zeroed. This register is loaded by the Rx Machine with the first received character. Reading the register causes the next register from the Rx FIFO to be loaded into RxD and RxF registers.

V_{IL}	Input Low Voltage	(1)	−0.5	0.8	V
V_{IH}	Input High Voltage	(1)	2.0	V_{CC}−0.5	V
V_{OL}	Output Low Voltage	(2), (9)		0.45	V
V_{OH}	Output High Voltage	(3), (9)	2.4		V
I_{LI}	Input Leakage Current	(4)		±10	μA
I_{LO}	3-State Leakage Current	(5)		±10	μA
I_{cc}	Power Supply Current	(6)		3.8	mA/MHz
I_{pd}	Power Down Supply	(7)		2	mA
I_{STBY}	Standby Supply Current	(10)		500	μA
I_{OHR}	$\overline{RTS}$, $\overline{DTR}$ Strapping Current	(11)		0.4	mA
I_{OLR}	$\overline{RTS}$, $\overline{DTR}$ Strapping Current	(12)	11		mA
C_{in}	Input Capacitance	(8)		10	pF
C_{io}	I/O Capacitance	(8)		10	pF
C_{XTAL}	X1, X2 Load			10	pF

NOTES:
1. Does not apply to CLK/X1 pin, when configured as crystal oscillator input (X1).
2. @ I_{OL} = 2 mA.
3. @ I_{OH} = −0.4 mA.
4. $0 < V_{IN} < V_{CC}$.
5. $0.45V < V_{OUT} < (V_{CC} - 0.45)$.
6. V_{CC} = 5.5V; V_{IL} = 0.5V (max); $V_{IH} = V_{CC} - 0.5V$ (min); 35 mA (max); Typical value = 2.5 mA/MHz (Not Tested); Ext 1X CLK (9 MHz max); $I_{OL} = I_{OH} = 0$.
7. V_{CC} = 5.5V; V_{IL} = GND; $V_{IH} = V_{CC}$; $I_{OL} = I_{OH} = 0$; device at power down mode, clock running.
8. Freq = 1 MHz.
9. Does not apply to OUT2/X2 pin, when configured as crystal oscillator output (X2).
10. Same as 7, but input clock not running.
11. Applies only during hardware reset for clock configuration options. Strapping current for logic HIGH.
12. Applies only during hardware reset for clock configuration. Strapping current for logic LOW.

A.C. SPECIFICATIONS

Testing Conditions:

- All AC output parameters are under output load of 20 to 100 pF, unless otherwise specified.
- AC testing inputs are driven at 2.4 for logic '1', and 0.45V for logic '0'. Output timing measurements are made at 1.5V for both a logical '0' and '1'.
- In the following tables, the units are ns, unless otherwise specified.

System Interface Specification—System Clock Specification:

The 82510 system clock is supplied via the CLK pin or generated by an on-chip crystal oscillator. The clock is optionally divided by two. The CLK parameters are given separately for internal divide-by-two option ACTIVE and INACTIVE.

The system clock (after division by two, if active) must be at least 16X the Tx or Rx baud rate (the faster of the two).

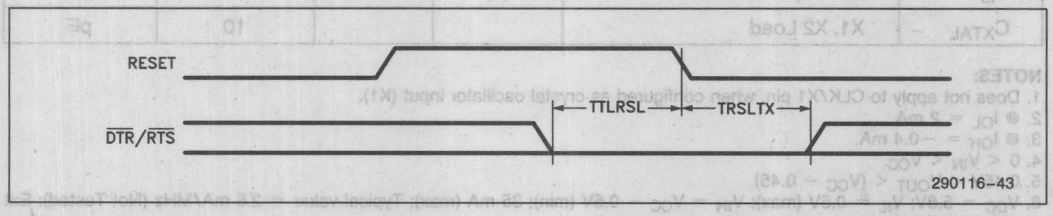
SYSTEM CLOCK SPECIFICATIONS

Symbol	Parameter	Min	Max	Notes
DIVIDE BY TWO OPTION—ACTIVE				
Tcy/2	CLK Period	54	250	(2)
TCLCH	CLK Low Time	25		
TCHCL	CLK High Time	25		
TCH1CH2	CLK Rise Time		10	(1)
TCL2CL1	CLK Fall Time		10	(1)
FXTAL	External Crystal Frequency Rating	4.0	18.432 MHz	
DIVIDE BY TWO OPTION—INACTIVE				
Tcy	CLK Period	108		
TCLCH	CLK Low Tme	54		
TCHCL	CLK High Time	44	250	
TCH1CH2	CLK Rise Time		15	(1)
TCL2CL1	CLK Fall Time		15	(1)

NOTES:
1. Rise/fall times are measured between 0.8 and 2.0V.
2. Tcy in ACTIVE divide by two option is TWICE the input clock period.

RESET SPECIFICATION

Symbol	Parameter	Min	Max	Notes
TRSHL	Reset Width—CLK/X1 Configured to CLK	8 Tcy		(1)
TTLRSL	RTS/DTR LOW Setup to Reset Inactive	6 Tcy		(2)
TRSLTX	RTS/DTR Low Hold after Reset Inactive	0	Tcy − 20	(2)

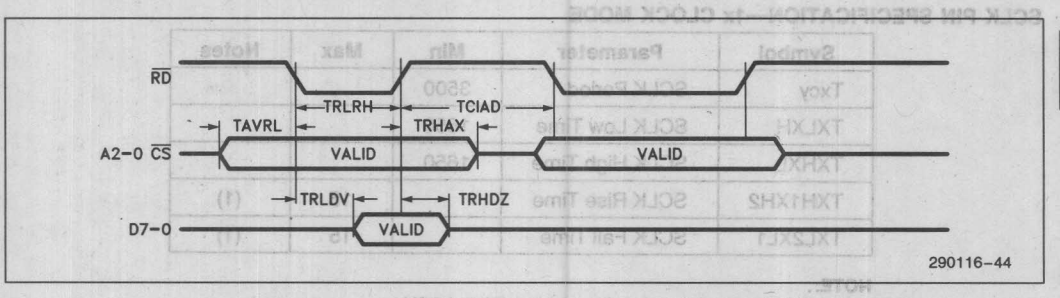


NOTES:
1. In case of CLK/X1 configured as X1, 1 Ms is required to guarantee crystal oscillator wake-up.
2. RTS/DTR are internally driven HIGH during RESET active time. The pin should be either left OPEN or externally driven LOW during RESET according to the required configuration of the system clock. These parameters specify the timing requirements on these pins, in case they are externally driven LOW during RESET.
The maximum spec on TRSLTX requires that the RTS/DTR pins not be forced later than TRSLTX maximum.

READ CYCLE SPECIFICATIONS

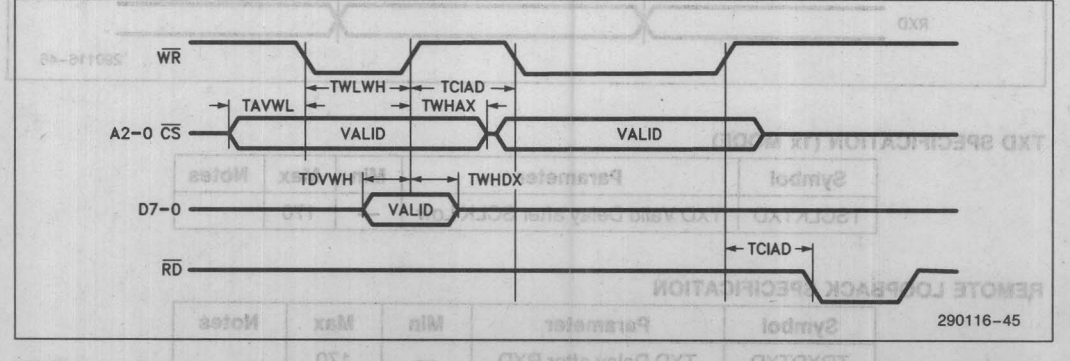
Symbol	Parameter	Min	Max	Notes
TRLRH	$\overline{RD}$ Active Width	2 Tcy + 65		
TAVRL	Address/$\overline{CS}$ Setup Time to $\overline{RD}$ Active	7		
TRHAX	Address/$\overline{CS}$ Hold Time after $\overline{RD}$ Inactive	0		
TRLDV	Data Out Valid Delay after $\overline{RD}$ Active		2Tcy + 65	
TCIAD	Command Inactive to Active Delay	Tcy + 15		(1)
TRHDZ	Data Out Float Delay after $\overline{RD}$ Inactive		40	

NOTE:
1. Command refers to either Read or Write signals.



WRITE CYCLE SPECIFICATION

Symbol	Parameter	Min	Max	Notes
TWLWH	$\overline{WR}$ Active Width	2Tcy + 15		
TAVWL	Address $\overline{CS}$ Setup Time to $\overline{WR}$ Active	7		
TWHAX	Address and $\overline{CS}$ Hold Time after $\overline{WR}$	0		
TDVWH	Data in Setup Time to $\overline{WR}$ Inactive	90		
TWHDX	Data in Hold Time after $\overline{WR}$ Inactive	12		



NOTE:
Many of the serial interface pins have more than one function; sometimes the different functions have different timings. In such a case, the timing of each function of a pin is given separately.

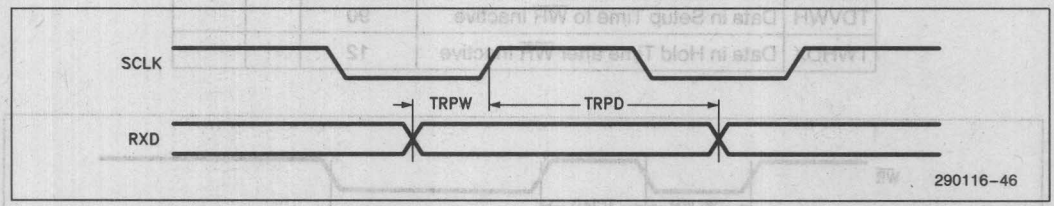
SCLK PIN SPECIFICATION—1x CLOCK MODE

Symbol	Parameter	Min	Max	Notes
Txcy	SCLK Period	3500		
TXLXH	SCLK Low Time	1650		
TXHXL	SCLK High Time	1650		
TXH1XH2	SCLK Rise Time		15	(1)
TXL2XL1	SCLK Fall Time		15	(1)

NOTE:
1. Rise/fall times are measured between 0.8V and 2.0V.

RXD SPECIFICATION (1x MODE)

Symbol	Parameter	Min	Max	Notes
TRPW	RXD Setup Time to SCLK High	250		
TRPD	RXD Hold Time After SCLK High	250		

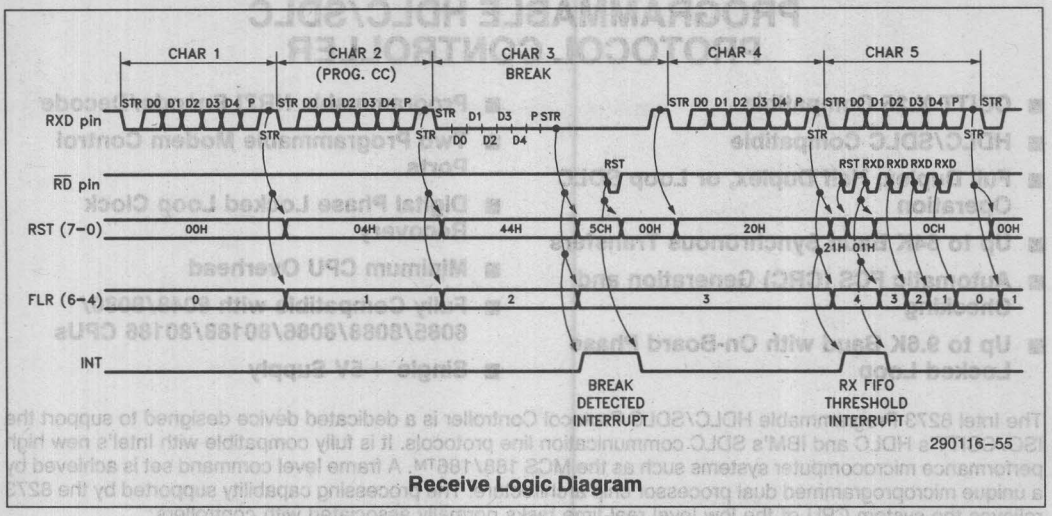


TXD SPECIFICATION (1x MODE)

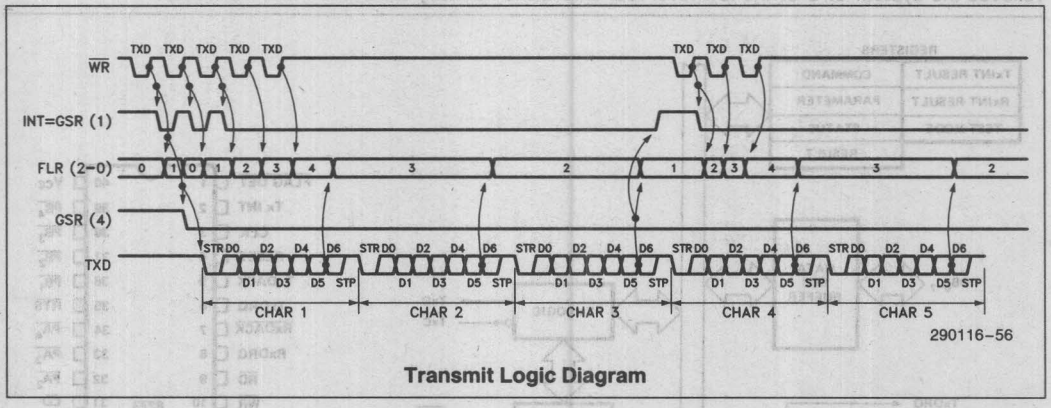
Symbol	Parameter	Min	Max	Notes
TSCLKTXD	TXD Valid Delay after SCLK Low	—	170	

REMOTE LOOPBACK SPECIFICATION

Symbol	Parameter	Min	Max	Notes
TRXDTXD	TXD Delay after RXD	—	170	



Receive Logic Diagram



Transmit Logic Diagram

8273
PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

- CCITT X.25 Compatible
- HDLC/SDLC Compatible
- Full Duplex, Half Duplex, or Loop SDLC Operation
- Up to 64K Baud Synchronous Transfers
- Automatic FCS (CRC) Generation and Checking
- Up to 9.6K Baud with On-Board Phase Locked Loop
- Programmable NRZI Encode/Decode
- Two Programmable Modem Control Ports
- Digital Phase Locked Loop Clock Recovery
- Minimum CPU Overhead
- Fully Compatible with 8048/8080/8085/8088/8086/80188/80186 CPUs
- Single +5V Supply

The Intel 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CCITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS 188/186™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

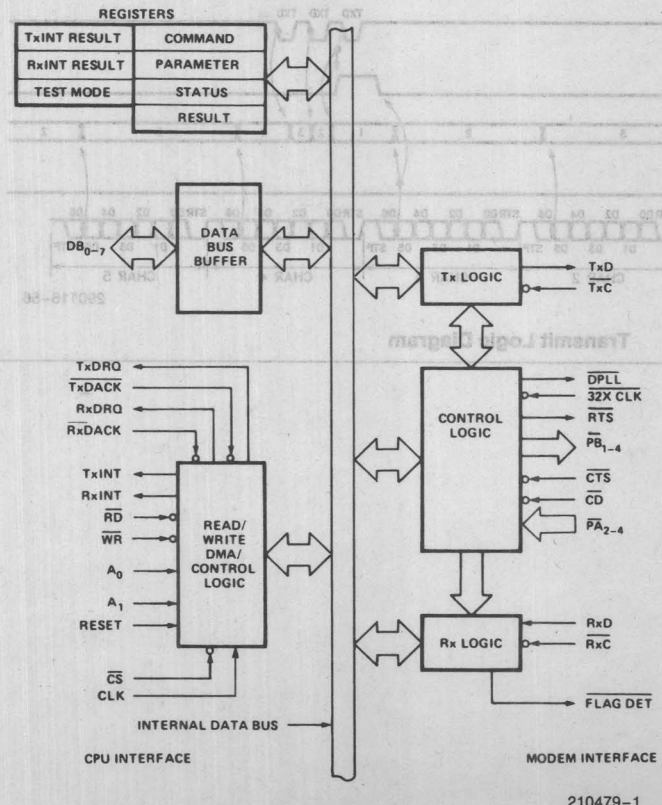


Figure 1. Block Diagram

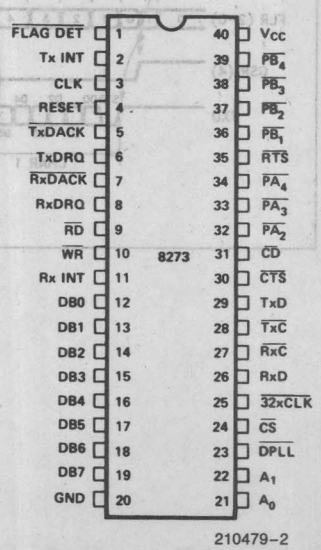


Figure 2. Configuration

December 1986
Order Number: 210479-003

A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame preambles called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more than fifteen consecutive ones, the station goes into an IDLE state.

Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

Frame Characteristics

An important characteristic of a frame is that is contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system—it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.

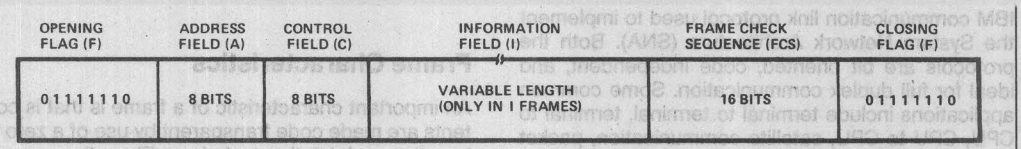


Figure 3. Frame Format

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
V_{CC}	40		**POWER SUPPLY:** +5V Supply.
GND	20		**GROUND:** Ground.
RESET	4	I	**RESET:** A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.
$\overline{CS}$	24	I	**CHIP SELECT:** The RD and WR inputs are enabled by the chip select input.
DB_0–DB_7	12–19	I/O	**DATA BUS:** The Data Bus lines are bidirectional three-state lines which interface with the system Data Bus.
$\overline{WR}$	10	I	**WRITE INPUT:** The Write signal is used to control the transfer of either a command or data from CPU to the 8273.
$\overline{RD}$	9	I	**READ INPUT:** The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.
TxINT	2	O	**TRANSMITTER INTERRUPT:** The Transmitter interrupt signal indicates that the transmitter logic requires service.
RxINT	11	O	**RECEIVER INTERRUPT:** The Receiver interrupt signal indicates that the Receiver logic requires service.
TxDRQ	6	O	**TRANSMITTER DATA REQUEST:** Requests a transfer of data between memory and the 8273 for a transmit operation.
RxRDQ	8	O	**RECEIVER DMA REQUEST:** Requests a transfer of data between the 8273 and memory for a receive operation.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{TXDACK}$	5	I	**TRANSMITTER DMA ACKNOWLEDGE:** The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.
$\overline{RxDACK}$	7	I	**RECEIVER DMA ACKNOWLEDGE:** The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.
A_0–A_1	21–22	I	**ADDRESS:** These two lines are CPU Interface Register Select lines.
TxD	29	O	**TRANSMITTER DATA:** This line transmits the serial data to the communication channel.
$\overline{TxC}$	28	I	**TRANSMITTER CLOCK:** The transmitter clock is used to synchronize the transmit data.
RxD	26	I	**RECEIVER DATA:** This line receives serial data from the communication channel.
$\overline{RxC}$	27	I	**RECEIVER CLOCK:** The Receiver Clock is used to synchronize the receive data.
$\overline{32X\ CLK}$	25	I	**32X CLOCK:** The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used.)
$\overline{DPLL}$	23	O	**DIGITAL PHASE LOCKED LOOP:** Digital Phase Locked Loop output can be tied to RxC and/or TxC when 1X clock is not available. DPLL is used with 32X CLK.
$\overline{FLAG\ DET}$	1	O	**FLAG DETECT:** Flag Detect signals that a flag (01111110) has been received by an active receiver.
$\overline{RTS}$	35	O	**REQUEST TO SEND:** Request to Send signals that the 8273 is ready to transmit data.
$\overline{CTS}$	30	I	**CLEAR TO SEND:** Clear to Send signals that the modem is ready to accept data from the 8273.
$\overline{CD}$	31	I	**CARRIER DETECT:** Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.
$\overline{PA}_{2-4}$	32–34	I	**GENERAL PURPOSE INPUT PORTS:** The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.
$\overline{PB}_{1-4}$	36–39	O	**GENERAL PURPOSE OUTPUT PORTS:** The CPU can write these output lines through Data Bus Buffer.
CLK	3	I	**CLOCK:** A square wave TTL clock.

2

FUNCTIONAL DESCRIPTION

General

The Intel 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications.

In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit. The 8273 recognizes and can generate flags (01111110) Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

CPU Interface

The CPU interface is optimized for the MCS-80/85™ bus with an 8257 DMA controller. However, the interface is flexible, and allows either DMA or non-DMA data transfers, interrupt or non-interrupt driven. It further allows maximum line utilization by providing early interrupt mechanism for buffered (only the information field can be transferred to memory) Tx command overlapping. It also provides separate Rx and Tx interrupt output channels for efficient operation. The 8273 keeps the interrupt request active until all the associated interrupt results have been read.

The CPU utilizes the CPU interface to specify commands and transfer data. It consists of seven registers addressed via $\overline{CIA}$, A_1, A_0, $\overline{RD}$ and $\overline{WR}$ signals and two independent data registers for receive data and transmit data. A_1, A_0 are generally derived from two low order bits of the address bus. If an 8080 based CPU is utilized, the $\overline{RD}$ and $\overline{WR}$ signals may be driven by the 8228 $\overline{I/OR}$ and $\overline{I/OW}$. The table shows the seven register select decoding:

A_1	A_0	$\overline{TxDACK}$	RxDACK	CS	$\overline{RD}$	$\overline{WR}$	Register
0	0	1	1	0	1	0	Command
0	0	1	1	0	0	1	Status
0	1	1	1	0	1	0	Parameter
0	1	1	1	0	0	1	Result
1	0	1	1	0	1	0	Reset
1	0	1	1	0	0	1	TxINT Result
1	1	1	1	0	1	0	—
1	1	1	1	0	0	1	RxINT Result
X	X	0	1	1	1	0	Transmit Data
X	X	1	0	1	0	1	Receive Data

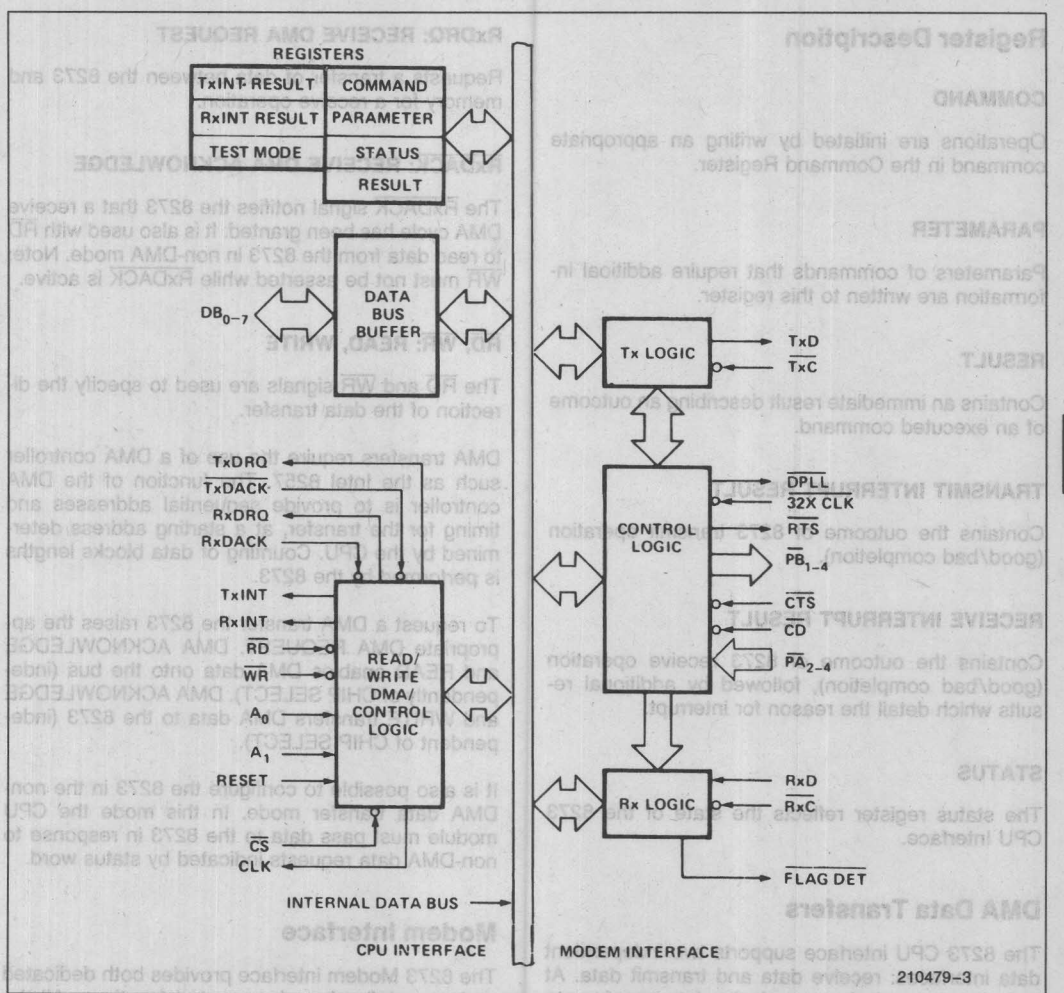


Figure 4. 8273 Block Diagram Showing CPU Interface Functions

2

Register Description

COMMAND

Operations are initiated by writing an appropriate command in the Command Register.

PARAMETER

Parameters of commands that require additioal information are written to this register.

RESULT

Contains an immediate result describing an outcome of an executed command.

TRANSMIT INTERRUPT RESULT

Contains the outcome of 8273 transmit operation (good/bad completion).

RECEIVE INTERRUPT RESULT

Contains the outcome of 8273 receive operation (good/bad completion), followed by additional results which detail the reason for interrupt.

STATUS

The status register reflects the state of the 8273 CPU Interface.

DMA Data Transfers

The 8273 CPU interface supports two independent data interfaces: receive data and transmit data. At high data transmission speeds the data transfer rate of the 8273 is great enough to justify the use of direct memory access (DMA) for the data transfers. When the 8273 is configured in DMA mode, the elements of the DMA interfaces are:

TxDRQ: TRANSMIT DMA REQUEST

Requests a transfer of data between memory and the 8273 for a transmit operation.

$\overline{\text{TxDACK}}$: TRANSMIT DMA ACKNOWLEDGE

The $\overline{\text{TxDACK}}$ signal notifies the 8273 that a transmit DMA cycle has been granted. It is also used with $\overline{\text{WR}}$ to transfer data to the 8273 in non-DMA mode. Note: $\overline{\text{RD}}$ must not be asserted while TxDACK is active.

RxDRQ: RECEIVE DMA REQUEST

Requests a transfer of data between the 8273 and memory for a receive operation.

$\overline{\text{RxDACK}}$: RECEIVE DMA ACKNOWLEDGE

The $\overline{\text{RxDACK}}$ signal notifies the 8273 that a receive DMA cycle has been granted. It is also used with $\overline{\text{RD}}$ to read data from the 8273 in non-DMA mode. Note: $\overline{\text{WR}}$ must not be asserted while $\overline{\text{RxDACK}}$ is active.

$\overline{\text{RD}}$, $\overline{\text{WR}}$: READ, WRITE

The $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer, at a starting address determined by the CPU. Counting of data blocks lengths is performed by the 8273.

To request a DMA transfer the 8273 raises the appropriate DMA REQUEST. DMA ACKNOWLEDGE and READ enables DMA data onto the bus (independently of CHIP SELECT). DMA ACKNOWLEDGE and WRITE transfers DMA data to the 8273 (independent of CHIP SELECT).

It is also possible to configure the 8273 in the non-DMA data transfer mode. In this mode the CPU module must pass data to the 8273 in response to non-DMA data requests indicated by status word.

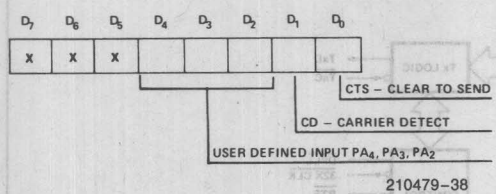
Modem Interface

The 8273 Modem interface provides both dedicated and user defined modem control functions. All the control signals are active low so that EIA RS-232C inverting drivers (MC 1488) and inverting receivers (MC 1489) may be used to interface to standard modems. For asynchronous operation, this interface supports programmable NRZI data encode/decode, a digital phase locked loop for efficient clock extraction from NRZI data, and modem control ports with automatic $\overline{\text{CTS}}$, $\overline{\text{CD}}$ monitoring and $\overline{\text{RTS}}$ generation. This interface also allows the 8273 to operate in PRE-FRAME SYNC mode in which the 8273 prefixes 16 transitions to a frame to synchronize idle lines before transmission of the first flag.

It should be noted that all the 8273 port operations deal with logical values, for instance, bit D0 of Port A will be a one when $\overline{\text{CTS}}$ (Pin 30) is a physical zero (logical one).

PORT A — INPUT PORT

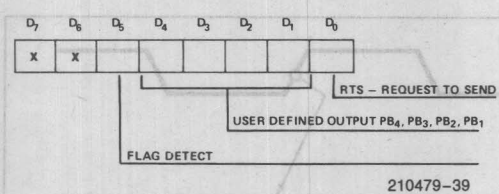
During operation, the 8273 interrogates input pins $\overline{CTS}$ (Clear to Send) and $\overline{CD}$ (Carrier Detect). $\overline{CTS}$ is used to condition the start of a transmission. If during transmission $\overline{CTS}$ is lost the 8273 generates an interrupt. During reception, if $\overline{CD}$ is lost, the 8273 generates an interrupt.



The user defined input bits correspond to the 8273 PA_4, PA_3 and PA_2 pins. The 8273 does not interrogate or manipulate these bits.

PORT B - OUTPUT PORT

During normal operation, if the CPU sets $\overline{RTS}$ active, the 8273 will not change this pin; however, if the CPU sets $\overline{RTS}$ inactive, the 8273 will activate it before each transmission and deactivate it one byte time after transmission. While the receiver is active the flag detect pin is pulsed each time a flag sequence is detected in the receive data stream. Following an 8273 reset, all pins of Port B are set to a high, inactive level.



The user defined output bits correspond to the state of PB_4–PB_1 pins. The 8273 does not interrogate or manipulate these bits.

Serial Data Logic

The Serial data is synchronized by the user transmit ($\overline{TxC}$) and receive ($\overline{RxC}$) clocks. The leading edge of $\overline{TxC}$ generates new transmit data and the trailing edge of $\overline{RxC}$ is used to capture receive data. The NRZI encoding/decoding of the receive and transmit data is programmable.

The diagnostic features included in the Serial Data logic are programmable loop back of data and selectable clock for the receiver. In the loop-back mode, the data presented to the TxD pin is internally routed to the receive data input circuitry in place of the RxD pin, thus allowing a CPU to send a message to itself to verify operation of the 8273.

In the selectable clock diagnostic feature, when the data is looped back, the receiver may be presented incorrect sample timing by the external circuitry. The user may select to substitute the $\overline{TxC}$ pin for the $\overline{RxC}$ input on-chip so that the clock used to generate the loop back data is used to sample it. Since TxD is generated off the leading edge of $\overline{TxC}$ and RxD is sampled on the trailing edge, the selected clock allows bit synchronism.

ASYNCHRONOUS MODE INTERFACE

Although the 8273 is fully compatible with the HDLC/SDLC communication line protocols, which are primarily designed for sychronous communication, the 8273 can also be used in asynchronous applications by using this interface. The interface employs a digital phase locked loop (DPLL) for clock recovery from a receive data stream and programmable NRZI encoding and decoding of data. The use of NRZI coding with SDLC transmission guarantees that within a frame, data transitions will occur at least every five bit times—the longest sequence of ones which may be transmitted without zero-bit insertion. The DPLL should be used only when NRZI coding is used since the NRZI coding will transmit zero sequence as line transitions. The digital phase locked loop also facilitates full-duplex and half-duplex asynchronous implementation with, or without modems.

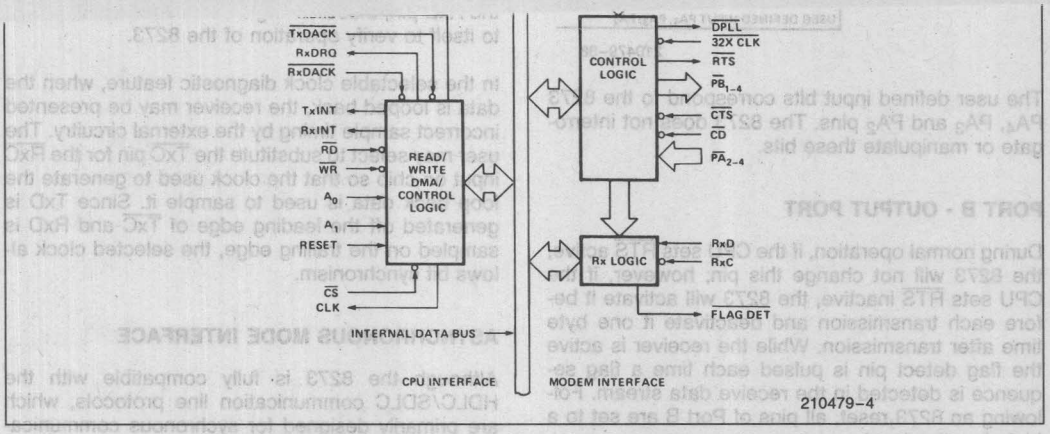


Figure 5. 8273 Block Diagram Showing Control Logic Functions

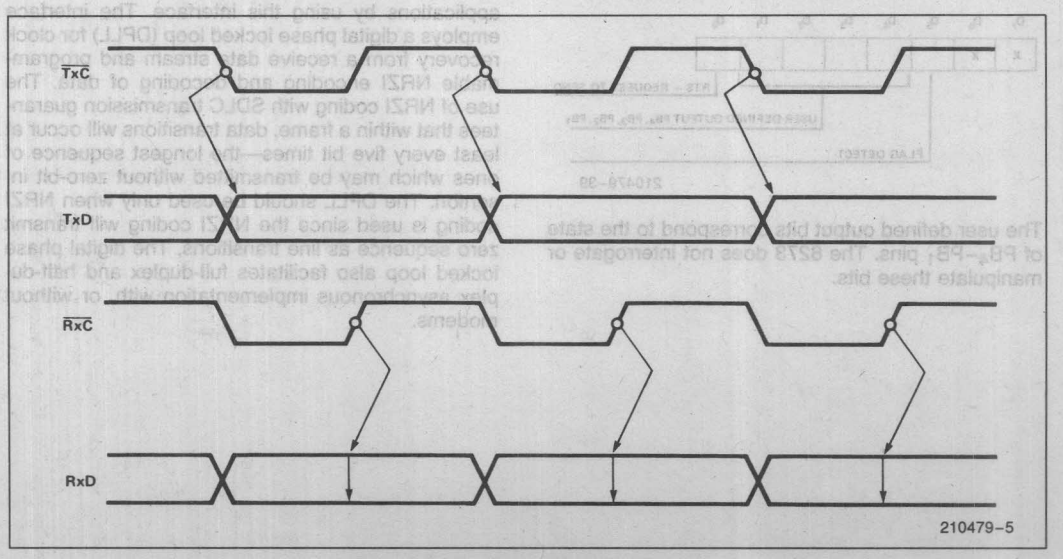


Figure 6. Transmit/Receive Timing

DIGITAL PHASE LOCKED LOOP

In asynchronous applications, the clock is derived from the receiver data stream by the use of the digital phase locked loop (DPLL). The DPLL requires a clock input at 32 times the required baud rate. The receive data (RxD) is sampled with this $\overline{32X\ CLK}$ and the 8273 DPLL supplies a sample pulse nominally centered on the RxD bit cells. The DPLL has a built-in "stiffness" which reduces sensitivity to line noise and bit distortion. This is accomplished by making phase error adjustments in discrete increments. Since the nominal pulse is made to occur at 32 counts of the $\overline{32X\ CLK}$, these counts are subtracted or added to the nominal, depending upon which quadrant of the four error quadrants the data edge occurs in. For example if an RxD edge is detected in quadrant A1, it is apparent that the $\overline{DPLL}$ sample "A" was placed too close to the trailing edge of the data cell; sample "B" will then be placed at T = ($T_{nominal}$ - 2 counts) = 30 counts of the $\overline{32X\ CLK}$ to move the sample pulse "B" toward the nominal center of the next bit cell. A data edge occuring in quadrant B1 would cause a smaller adjustment of phase with T = 31 counts of the $\overline{32X\ CLK}$. Using this technique the $\overline{DPLL}$ pulse will converge to nominal bit center within 12 data bit times, worst case, with constant incoming RxD edges.

A method of attaining bit synchronism following a line idle is to use PRE-FRAME SYNC mode of transmission.

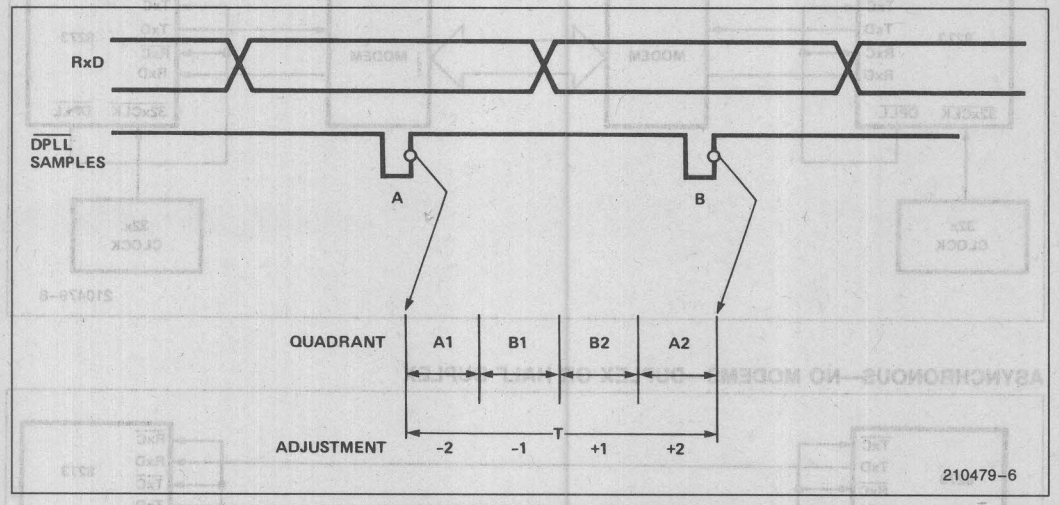
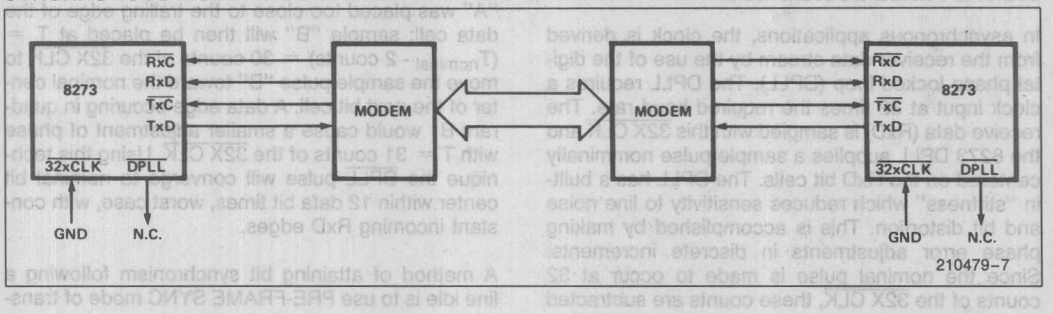


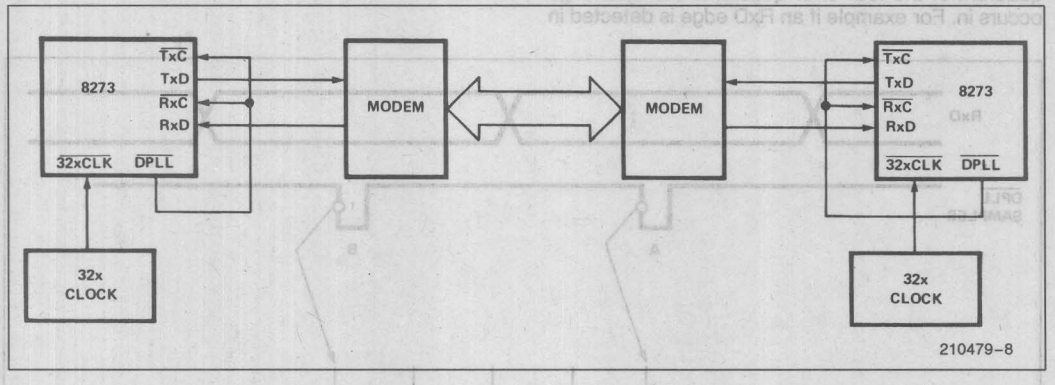
Figure 7. DPLL Sample Timing

SYNCHRONOUS MODEM—DUPLEX OR HALF DUPLEX OPERATION



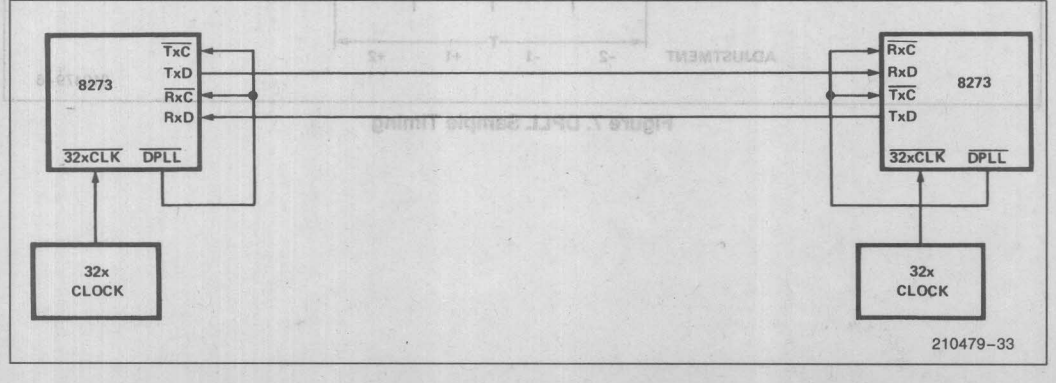
210479-7

ASYNCHRONOUS MODES—DUPLEX OR HALF DUPLEX OPERATION



210479-8

ASYNCHRONOUS—NO MODEMS—DUPLEX OR HALF DUPLEX



210479-33

SDLC LOOP

The DPLL simplifies the SDLC loop station implementation. In this application, each secondary station on a loop data link is a repeater set in one-bit delay mode. The signals sent out on the loop by the loop controller (primary station) are relayed from station to station then, back to the controller. Any secondary station finding its address in the A field captures the frame for action at that station. All received frames are relayed to the next station on the loop.

Loop stations are required to derive bit timing from the incoming NRZI data stream. The DPLL generates sample Rx clock timing for reception and uses the same clock to implement Tx clock timing.

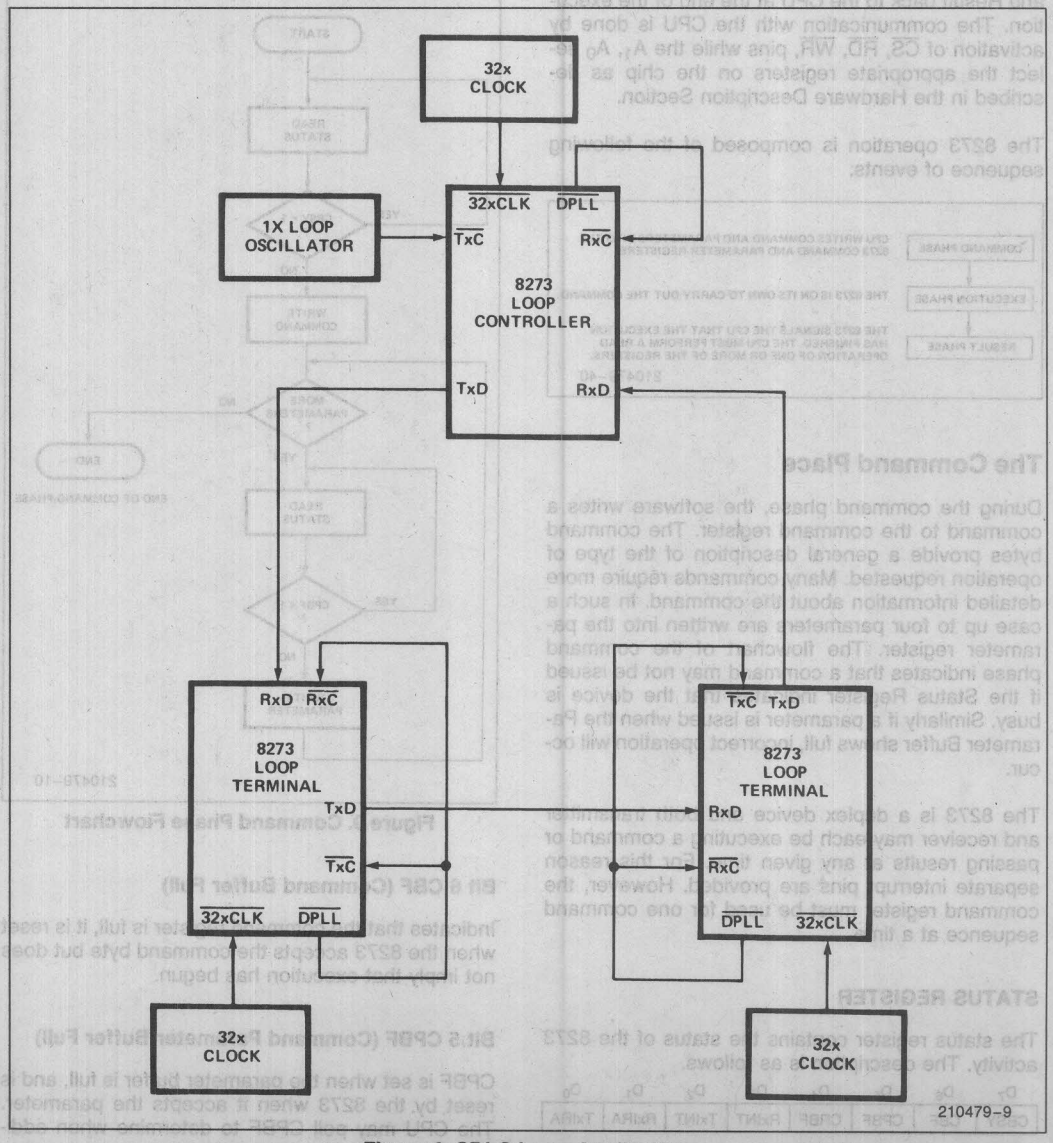
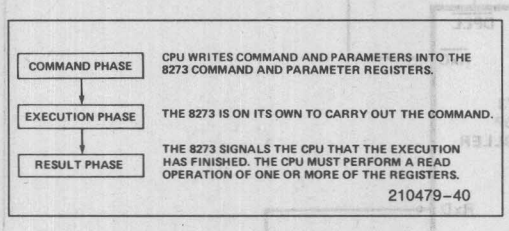


Figure 8. SDLC Loop Application

PRINCIPLES OF OPERATION

The 8273 is an intelligent peripheral controller which relieves the CPU of many of the rote tasks associated with constructing and receiving frames. It is fully compatible with the MCS-80/85™ system bus. As a peripheral device, it accepts commands from a CPU, executes these commands and provides an Interrupt and Result back to the CPU at the end of the execution. The communication with the CPU is done by activation of $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, pins while the A_1, A_0 select the appropriate registers on the chip as described in the Hardware Description Section.

The 8273 operation is composed of the following sequence of events:



The Command Place

During the command phase, the software writes a command to the command register. The command bytes provide a general description of the type of operation requested. Many commands require more detailed information about the command. In such a case up to four parameters are written into the parameter register. The flowchart of the command phase indicates that a command may not be issued if the Status Register indicates that the device is busy. Similarly if a parameter is issued when the Parameter Buffer shows full, incorrect operation will occur.

The 8273 is a duplex device and both transmitter and receiver may each be executing a command or passing results at any given time. For this reason separate interrupt pins are provided. However, the command register must be used for one command sequence at a time.

STATUS REGISTER

The status register contains the status of the 8273 activity. The description is as follows.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

Bit 7 CBSY (Command Busy)

Indicates in-progress command, set for CPU poll when Command Register is full, reset upon command phase completion. It is improper to write a command when CBSY is set; it results in incorrect operation.

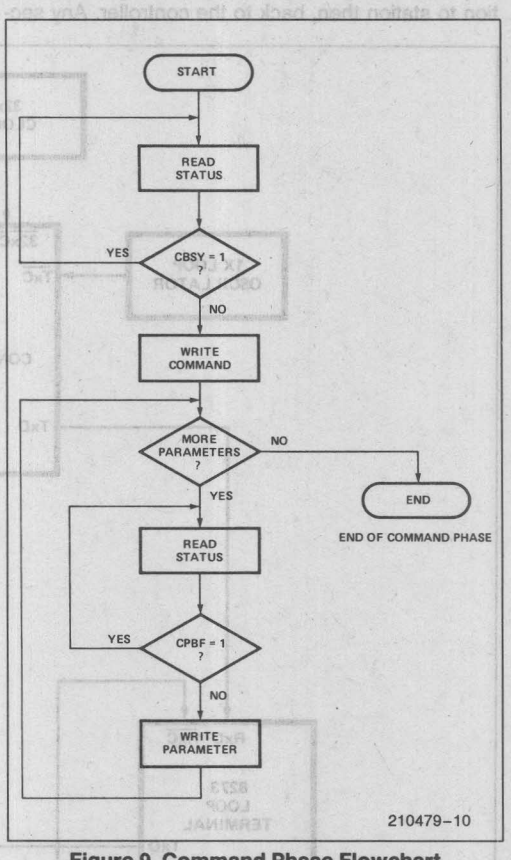


Figure 9. Command Phase Flowchart

Bit 6 CBF (Command Buffer Full)

Indicates that the command register is full, it is reset when the 8273 accepts the command byte but does not imply that execution has begun.

Bit 5 CPBF (Command Parameter Buffer Full)

CPBF is set when the parameter buffer is full, and is reset by the 8273 when it accepts the parameter. The CPU may poll CPBF to determine when additional parameters may be written.

Bit 4 CRBF (Command Result Buffer Full)

Indicates that an executed command immediate result is present in the Result Register. It is set by 8273 and reset when CPU reads the result.

Bit 3 RxINT (Receiver Interrupt)

RxINT indicates that the receiver requires CPU attention. It is identical to RxINT (pin 11) and is set by the 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has received a data byte from the 8273 in a Non-DMA data transfer.

Bit 2 TxINT (Transmitter Interrupt)

The TxINT indicates that the transmitter requires CPU attention. It is identical to TxINT (pin 2). It is set by 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has transferred transmit data byte to the 8273 in a Non-DMA transfer.

Bit 1 RxIRA (Receiver Interrupt Result Available)

The RxIRA is set by the 8273 when an interrupt result byte is placed in the RxINT register. It is reset after the CPU has read the RxINT register.

Bit 0 TxIRA (Transmitter Interrupt Result Available)

The TxIRA is set by the 8273 when an interrupt result byte is placed in the TxINT register. It is reset when the CPU has read the TxINT register.

THE EXECUTION PHASE

Upon accepting the last parameter, the 8273 enters into the Execution Phase. The execution phase may consist of a DMA or other activity, and may or may not require CPU intervention. The CPU intervention is eliminated in this phase if the system utilizes DMA for the data transfers, otherwise, for non-DMA data transfers, the CPU is interrupted by the 8273 via TxINT and RxINT pins, for each data byte request.

2

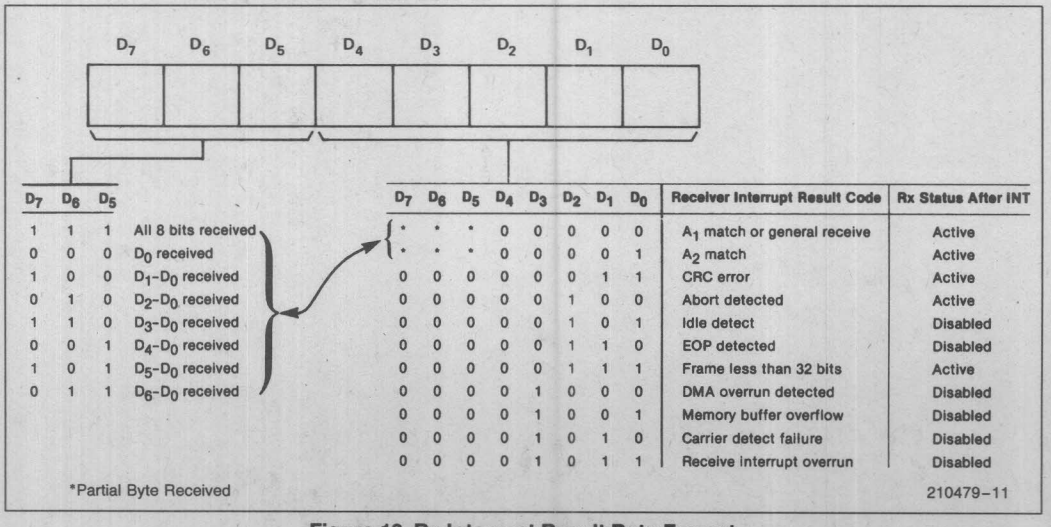
THE RESULT PHASE

During the result phase, the 8273 notifies the CPU of the execution outcome of a command. This phase is initiated by:

1. The successful completion of an operation
2. An error detected during an operation.

To facilitate quick network software decisions, two types of execution results are provided:

1. An Immediate Result
2. A Non-Immediate Result



D_7	D_6	D_5	
1	1	1	All 8 bits received
0	0	0	D_0 received
1	0	0	D_1-D_0 received
0	1	0	D_2-D_0 received
1	1	0	D_3-D_0 received
0	0	1	D_4-D_0 received
1	0	1	D_5-D_0 received
0	1	1	D_6-D_0 received

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Receiver Interrupt Result Code	Rx Status After INT
*	*	*	0	0	0	0	0	A_1 match or general receive	Active
*	*	*	0	0	0	0	1	A_2 match	Active
0	0	0	0	0	0	1	1	CRC error	Active
0	0	0	0	0	1	0	0	Abort detected	Active
0	0	0	0	0	1	0	1	Idle detect	Disabled
0	0	0	0	0	1	1	0	EOP detected	Disabled
0	0	0	0	0	1	1	1	Frame less than 32 bits	Active
0	0	0	0	1	0	0	0	DMA overrun detected	Disabled
0	0	0	0	1	0	0	1	Memory buffer overflow	Disabled
0	0	0	0	1	0	1	0	Carrier detect failure	Disabled
0	0	0	0	1	0	1	1	Receive interrupt overrun	Disabled

*Partial Byte Received

210479-11

Figure 10. Rx Interrupt Result Byte Format

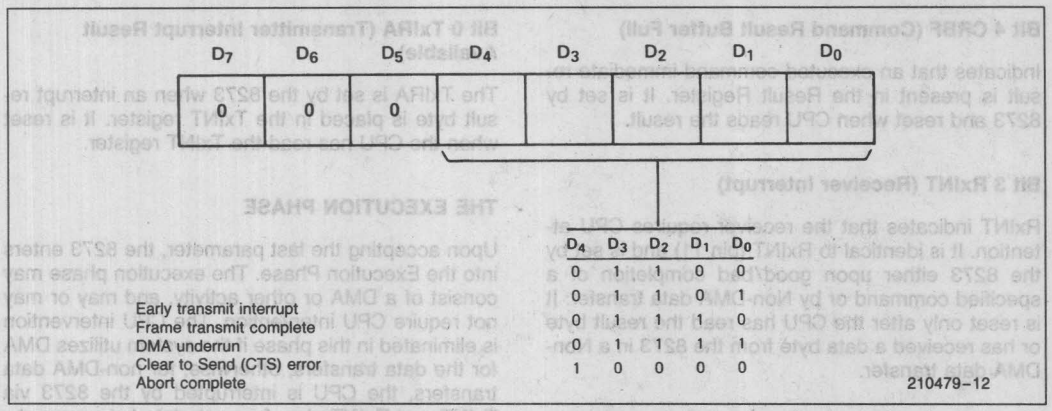


Figure 11. Tx Interrupt Result Byte Format

Immediate result is provided by the 8273 for commands such as Read Port A and Read Port B which have information ($\overline{CTS}$, $\overline{CD}$, $\overline{RTS}$, etc.) that the network software needs to make quick operational decisions.

A command which cannot provide an immediate result will generate an interrupt to signal the beginning of the Result phase. The immediate results are provided in the Result Register; all non-immediate results are available upon device interrupt, through Tx Interrupt Result Register TxI/R or Rx Interrupt Result Register RxI/R. The result may consist of a one-byte interrupt code indicating the condition for the interrupt and, if required, one or more bytes which detail the condition.

Tx and Rx Interrupt Result Registers

The Result Registers have a result code, the three high order bits D_7–D_5 of which are set to zero for all but the receive command. This command result contains a count that indicates the number of bits received in the last byte. If a partial byte is received, the high order bits of the last data byte are indeterminate.

All results indicated in the command summary must be read during the result phase.

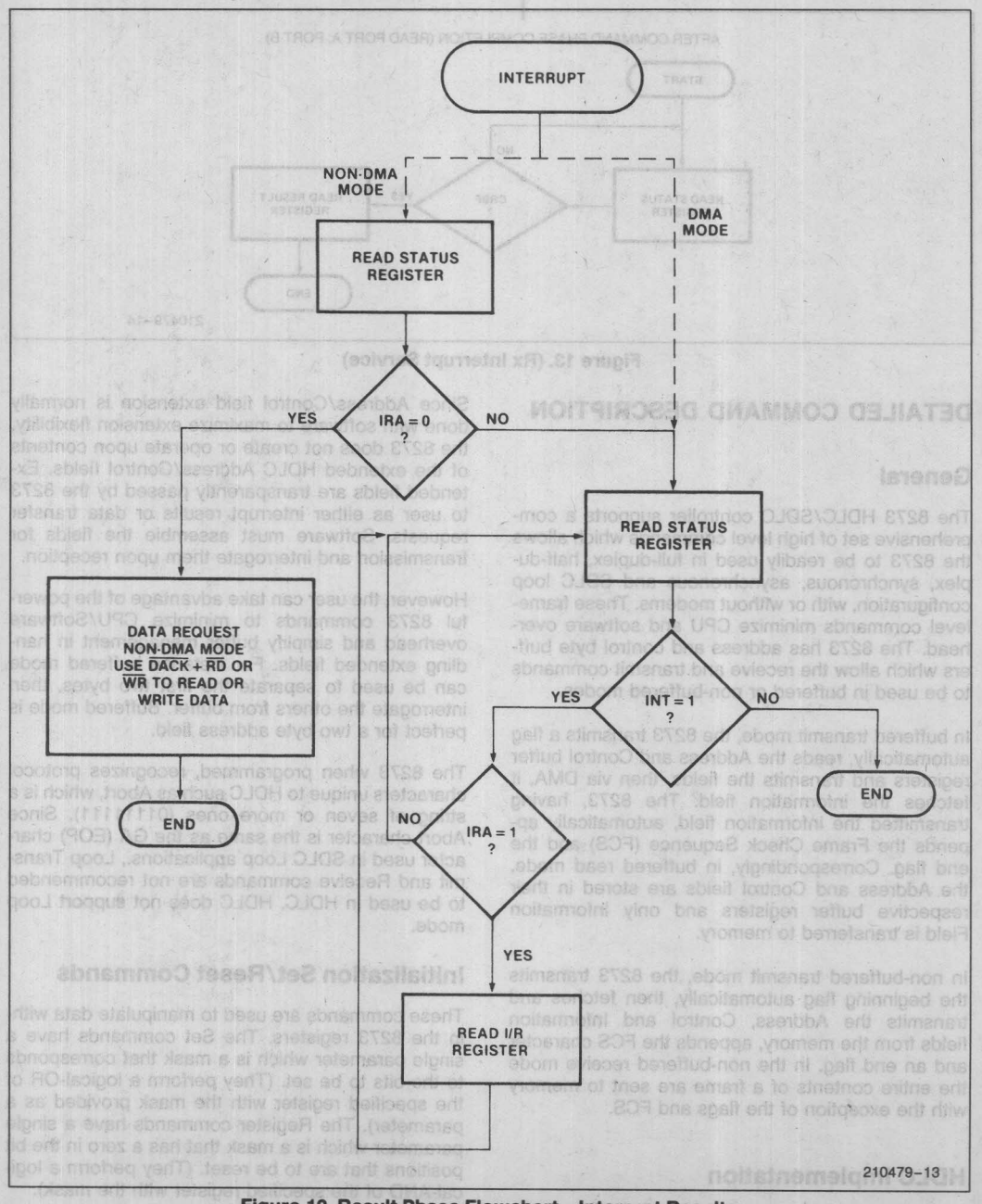


Figure 12. Result Phase Flowchart—Interrupt Results

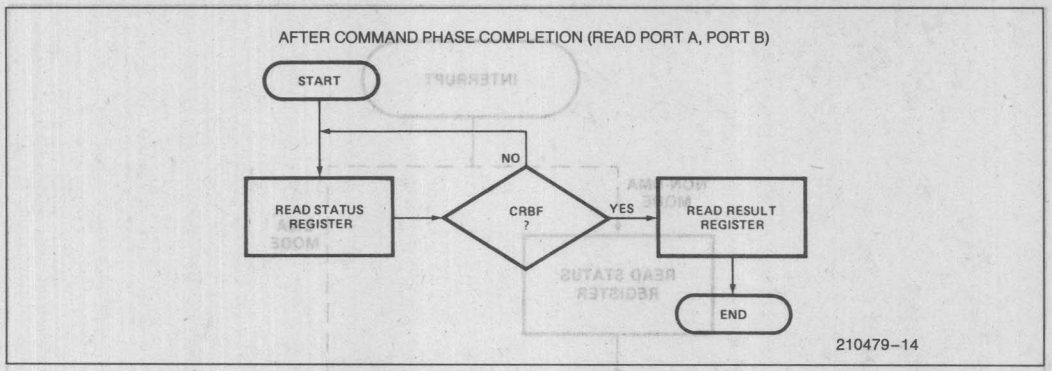


Figure 13. (Rx Interrupt Service)

DETAILED COMMAND DESCRIPTION

General

The 8273 HDLC/SDLC controller supports a comprehensive set of high level commands which allows the 8273 to be readily used in full-duplex, half-duplex, synchronous, asynchronous and SDLC loop configuration, with or without modems. These frame-level commands minimize CPU and software overhead. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In buffered transmit mode, the 8273 transmits a flag automatically, reads the Address and Control buffer registers and transmits the fields, then via DMA, it fetches the information field. The 8273, having transmitted the information field, automatically appends the Frame Check Sequence (FCS) and the end flag. Correspondingly, in buffered read mode, the Address and Control fields are stored in their respective buffer registers and only Information Field is transferred to memory.

In non-buffered transmit mode, the 8273 transmits the beginning flag automatically, then fetches and transmits the Address, Control and Information fields from the memory, appends the FCS character and an end flag. In the non-buffered receive mode the entire contents of a frame are sent to memory with the exception of the flags and FCS.

HDLC Implementation

HDLC Address and Control field are extendable. The extension is selected by setting the low order bit of the field to be extended to a one, a zero in the low order bit indicates the last byte of the respective *field.*

Since Address/Control field extension is normally done with software to maximize extension flexibility, the 8273 does not create or operate upon contents of the extended HDLC Address/Control fields. Extended fields are transparently passed by the 8273 to user as either interrupt results or data transfer requests. Software must assemble the fields for transmission and interrogate them upon reception.

However, the user can take advantage of the powerful 8273 commands to minimize CPU/Software overhead and simplify buffer management in handling extended fields. For instance buffered mode can be used to separate the first two bytes, then interrogate the others from buffer. Buffered mode is perfect for a two byte address field.

The 8273 when programmed, recognizes protocol characters unique to HDLC such as Abort, which is a string of seven or more ones (01111111). Since Abort character is the same as the GA (EOP) character used in SDLC Loop applications., Loop Transmit and Receive commands are not recommended to be used in HDLC. HDLC does not support Loop mode.

Initialization Set/Reset Commands

These commands are used to manipulate data within the 8273 registers. The Set commands have a single parameter which is a mask that corresponds to the bits to be set. (They perform a logical-OR of the specified register with the mask provided as a parameter). The Register commands have a single parameter which is a mask that has a zero in the bit positions that are to be reset. (They perform a logical-AND of the specified register with the mask).

SET ONE-BIT DELAY (CMD CODE A4)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	1	0	0	1	0	0
PAR:	0	1	1	0	0	0	0	0	0	0

When one bit delay is set, 8273 retransmits the received data stream one bit delayed. This mode is entered at a receiver character boundary, and should only be used by Loop Stations.

RESET ONE-BIT DELAY (CMD CODE 64)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	1	0	0	1	0	0
PAR:	0	1	0	1	1	1	1	1	1	1

The 8273 stops the one bit delayed retransmission mode.

SET DATA TRANSFER MODE (CMD CODE 97)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	0	1	0	1	1	1
PAR:	0	1	0	0	0	0	0	0	0	1

When the data transfer mode is set, the 8273 will interrupt when data bytes are required for transmission or are available from a receive. If a transmit interrupt occurs and the status indicates that there is no Transmit Result (TxIRA = 0), the interrupt is a transmit data request. If a receive interrupt occurs and the status indicates that there is no receive result (RxIRA = 0), the interrupt is a receive data request.

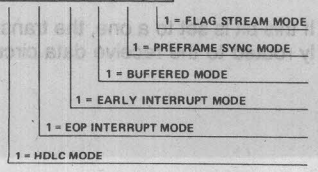
RESET DATA TRANSFER MODE (CMD CODE 57)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	0	1	0	1	1	1
PAR:	0	1	1	1	1	1	1	1	1	0

If the Data Transfer Mode is reset, the 8273 data transfers are performed through the DMA requests without interrupting the CPU.

SET OPERATING MODE (CMD CODE 91)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	0	1	0	0	0	1
PAR:	0	1	0	0						



210479-34

RESET OPERATING MODE (CMD CODE 51)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	0	1	0	0	0	1
PAR:	0	1	1	1						

Any mode switches set in CMD code 91 can be reset using this command by placing zeros in the appropriate positions.

(D5) HDLC MODE

In HDLC mode, a bit sequence of seven ones (01111111) is interpreted as as an abort character. Otherwise, eight ones (011111111) signal an abort.

(D4) EOP INTERRUPT MODE

In EOP interrupt mode, an interrupt is generated whenever an EOP character (01111111) is detected by an active receiver. This mode is useful for the implementation of an SDLC loop controller in detecting the end of a message stream after a loop poll.

(D3) TRANSMITTER EARLY INTERRUPT MODE (Tx)

The early interrupt mode is specified to indicate when the 8273 should generate an end of frame interrupt. When set, an early interrupt is generated when the last data character has been passed to the 8273. If the user software responds with another transmit command before the final flag is sent, the final flag interrupt will not be generated and a new frame will immediately begin when the current frame is complete. This permits frames to be separated by a single flag. If no additional Tx commands are provided, a final interrupt will follow.

NOTE:
In buffered mode, if a supervisory frame (no Information) Transmit command is sent in response to an early Transmit Interrupt, the 8273 will repeatedly transmit the same supervisory frame with one flag in between, until a non-supervisory transmit is issued.

Early transmitter interrupt can be used in buffered mode by waiting for a transmit complete interrupt instead of early Transmit Interrupt before issuing a transmit frame command for a supervisory frame. See Figure 14.

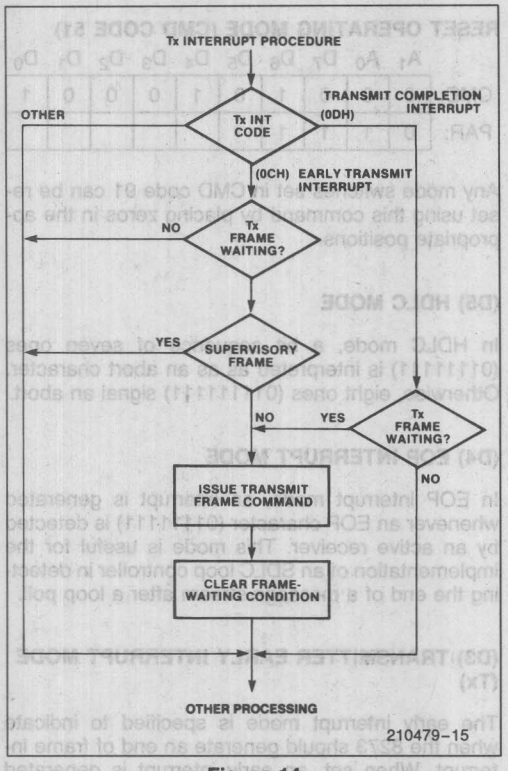


Figure 14

If this bit is zero, the interrupt will be generated only after the final flag has been transmitted.

(D2) BUFFERED MODE

If the buffered mode bit is set to a one, the first two bytes (normally the address (A) and control (C) fields) of a frame are buffered by the 8273. If this bit is a zero the address and control fields are passed to and from memory.

(D1) PREFRAME SYNC MODE

If this bit is set to a one the 8273 will transmit two characters before the first flag of a frame.

To guarantee sixteen line transitions, the 8273 sends two bytes of data $(00)_H$ if NRZI is set or data $(55)_H$ if NRZI is not set.

(D0) FLAG STREAM MODE

If this bit is set to a one, the following table outlines the operation of the transmitter.

Transmitter State	Action
Idle	Send Flags Immediately.
Transmit or Transmit Transparent Active	Send Flags After the Transmission Complete
Loop Transmit Active	Ignore Command.
1 Bit Delay Active	Ignore Command.

If this bit is reset to zero the following table outlines the operation of the transmitter

Transmitter State	Action
IDLE	Sends Idles on Next Character boundary.
Transmit or Transmit-Transparent Active	Send Idles after the Transmission is Complete.
Loop Transmit Active	Ignore Command.
1 Bit Delay Active	Ignore Command.

SET SERIAL I/O MODE (CMD CODE A0)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	0	1	0	0	0	0	0
PAR:	0	1	0	0	0	0	0			

1 = NRZ1 MODE
1 = TxC → RxC
1 = LOOP BACK TxD → RxD

210479-16

RESET SERIAL I/O MODE (CMD CODE 60)

This command allows bits set in CMD code A0 to be reset by placing zeros in the appropriate positions.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	1	1	0	0	0	0	0
PAR:	0	1	1	1	1	1	1			

(D2) LOOP BACK

If this bit is set to a one, the transmit data is internally routed to the receive data circuitry.

(D1) TxC → RxC

If this bit is set to a one, the transmit clock is internally routed to the receive clock circuitry. It is normally used with the loop back bit (D2).

(D0) NRZI MODE

If this bit is set to a one, NRZI encoding and decoding of transmit and receive data is provided. If this bit is a zero, the transmit and receive data is treated as a normal positive logic bit stream.

NRZI encoding specifies that a zero causes a change in the polarity of the transmitted signal and a one causes no polarity change. NRZI is used in all asynchronous operations. Refer to IBM document GA27-3093 for details.

Reset Device Command

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TMR:	1	0	0	0	0	0	0	0	0	1
TMR:	1	0	0	0	0	0	0	0	0	0

An 8273 reset command is executed by outputting a $(01)_H$ followed by $(00)_H$ to the reset register (TMR). See 8273 AC timing characteristics for Reset pulse specifications.

The reset command emulates the action of the reset pin.

1) The modem control signals are forced high (inactive level).
2) The 8273 status register flags are cleared.
3) Any commands in progress are terminated immediately.
4) The 8273 enters an idle state until the next command is issued.
5) The Serial I/O and Operating Mode registers are set to zero and DMA data register transfer mode is selected.
6) The device assumes a non-loop SDLC terminal role.

Receive Commands

The 8273 supports three receive commands: General Receive, Selective Receive, and Selective Loop Receive.

GENERAL RECEIVE (CMD CODE C0)

General receive is a receive mode in which frames are received regardless of the contents of the address field.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							

NOTES:

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receive is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.

2

SELECTIVE RECEIVE (CMD CODE C1)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective receive is a receive mode in which frames are ignored unless the address field matches any one of two address fields given to the 8273 as parameters.

When selective receive is used in HDLC the 8273 looks at the first character, if extended, software must then decide if the message is for this unit.

SELECTIVE LOOP RECEIVE (CMD CODE C2)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	0	1	0
PAR:	0	0	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective loop receive operates like selective receive except that the transmitter is placed in flag stream mode automatically after detecting an EOP (01111111) following a valid received frame. The one bit delay mode is also reset at the end of a selective loop receive.

RECEIVE DISABLE (CMD CODE 5)

Terminates an active receive command immediately.

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	0	1	0	1

PAR: NONE

Transmit Commands

The 8273 supports three transmit commands: Transmit Frame, Loop Transmit, Transmit Transparent.

TRANSMIT FRAME (CMD CODE C8)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame including: initial flag, frame check sequence, and the final flag.

If the buffered mode is specified, the L0, L1, frame length provides as a parameter is the length of the information field and the address and control fields must be input.

In unbuffered mode the frame length provided must be the length of the information field plus two and the address and control fields must be the first two bytes of data. Thus only the frame length bytes are required as parameters.

LOOP TRANSMIT (CMD CODE CA)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	1	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame in the same manner as the transmit frame command except:

1) If the flag stream mode is not active transmission will begin after a received EOP has been converted to a flag.
2) If the flag stream mode is active transmission will begin at the next flag boundary for buffered mode or at the third flag boundary for non-buffered mode.
3) At the end of a loop transmit the one-bit delay mode is entered and the flag stream mode is reset.

TRANSMIT TRANSPARENT (CMD CODED C9)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							

The 8273 will transmit a block of raw data without protocol, i.e., no zero bit insertion, flags, or frame check sequences.

Abort Transmit Commands

An abort command is supported for each type of transmit command. The abort commands are ignored if a transmit command is not in progress.

ABORT TRANSMIT FRAME (CMD CODE CC)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	0	0

PAR: NONE

After an abort character (eight contiguous ones) is transmitted, the transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

ABORT LOOP TRANSMIT (CMD CODE CE)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	1	0

PAR: NONE

After a flag is transmitted the transmitter reverts to one bit delay mode.

ABORT TRANSMIT TRANSPARENT (CMD CODE CD)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	1	1	0	0	1	1	0	1

PAR: NONE

The transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

Modem Control Commands

The modem control commands are used to manipulate the modem control ports.

When read Port A or Port B commands are executed the result of the command is returned in the result register. The Bit Set Port B command requires a parameter that is a mask that corresponds to the bits to be set. The Bit Reset Port B command requires a mask that has a zero in the bit positions that are to be reset.

READ PORT A (CMD CODE 22)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	0	0	0	1	0

PAR: NONE

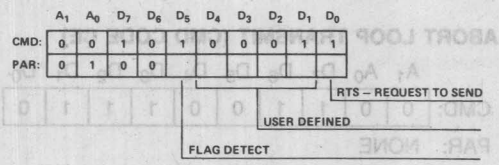
READ PORT B (CMD CODE 23)

	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CMD:	0	0	0	0	1	0	0	0	1	1

PAR: NONE

SET PORT B BITS (CMD CODE A3)

This command allows user defined Port B pins to be set.



210479-35

(D_5) FLAG DETECT

This bit can be used to set the flag detect pin. However, it will be reset when the next flag is detected.

(D_4–D_1) USER DEFINED OUTPUTS

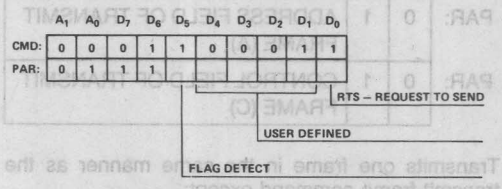
These bits correspond to the state of the PB_4–PB_1 output pins.

(D_0) REQUEST TO SEND

This is a dedicted 8273 modem control signal, and reflects the same logical state of RTS pin.

RESET PORT B BITS (CMD CODE 63)

This command allows Port B user defined bits to be reset.



210479-36

This command allows Port B (D_4–D_1) user defined bits to be reset. These bits correspond to Output Port pins (PB_4–PB_1).

8273 Command Summary

Command Description	Command HEX	Parameter	Results	Result Port	Completion Interrupt
Set One Bit Delay	A4	Set Mask	None	—	No
Reset One Bit Delay	64	Reset Mask	None	—	No
Set Data Transfer Mode	97	Set Mask	None	—	No
Reset Data Transfer Mode	57	Reset Mask	None	—	No
Set Operating Mode	91	Set Mask	None	—	No
Reset Operating Mode	51	Reset Mask	None	—	No
Set Serial I/O Mode	A0	Set Mask	None	—	No
Reset Serial I/O Mode	60	Reset Mask	None	—	No
General Receive	C0	B0, B1	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Selective Receive	C1	B0,B1,A1,A2	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Selective Loop Receive	C2	B0,B1,A1,A2	RIC,R0,R1,(A,C)(2)	RXI/R	Yes
Receive Disable	C5	None	None	—	No
Transmit Frame	C8	L0,L1,(A,C)(1)	TIC	TXI/R	Yes
Loop Transmit	CA	L0,L1,(A,C)(1)	TIC	TXI/R	Yes
Transmit Transparent	C9	L0,L1	TIC	TXI/R	Yes
Abort Transmit Frame	CC	None	TIC	TXI/R	Yes

8273 Command Summary (Continued)

Command Description	Command HEX	Parameter	Results	Result Port	Completion Interrupt
Abort Loop Transmit	CE	None	TIC	TXI/R	Yes
Abort Transmit Transparent	CD	None	TIC	TXI/R	Yes
Read Port A	22	None	Port Value	Result	No
Read Port B	23	None	Port Value	Result	No
Set Port B Bit	A3	Set Mask	None	—	No
Reset Port B Bit	63	Reset Mask	None	—	No

NOTES:
1. Issued only when in buffered mode.
2. Read as results only in buffered mode.

8273 Command Summary Key

B0— Least significant byte of the receiver buffer length.

B1— Most significant byte of the receive buffer length.

L0— Least significant byte of the Tx frame length.

L1— Most significant byte of the Tx frame length.

A1— Receive frame address match field one.

A2— Receive frame address match field two.

A— Address field of received frame. If non-buffered mode is specified, this result is not provided.

C— Control field of received frame. If non-buffered mode is specified this result is not provided.

RXI/R— Receive interrupt result register.

TXI/R— Transmit interrupt result register.

R0— Least significant byte of the length of the frame received.

R1— Most significant byte of the length of the frame received.

RIC— Receiver interrupt result code.

TIC— Transmitter interrupt result code.

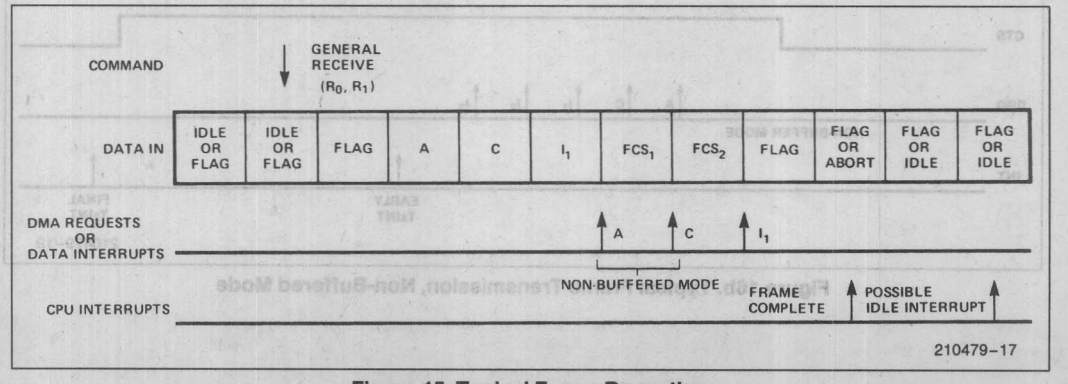


Figure 15. Typical Frame Reception

NOTE:
In order to ensure proper operation to the maximum baud rate, Receive commands or Read/Write Port commands should be written only when either the transmitter or the receiver is inactive. In full duplex systems, it is recommended that these commands be issued after servicing a transmitter interrupt but before a new transmit command is issued. When operating in full Duplex (active transmitter or receiver) with commands, the maximum data rate decreases to 49K Baud.

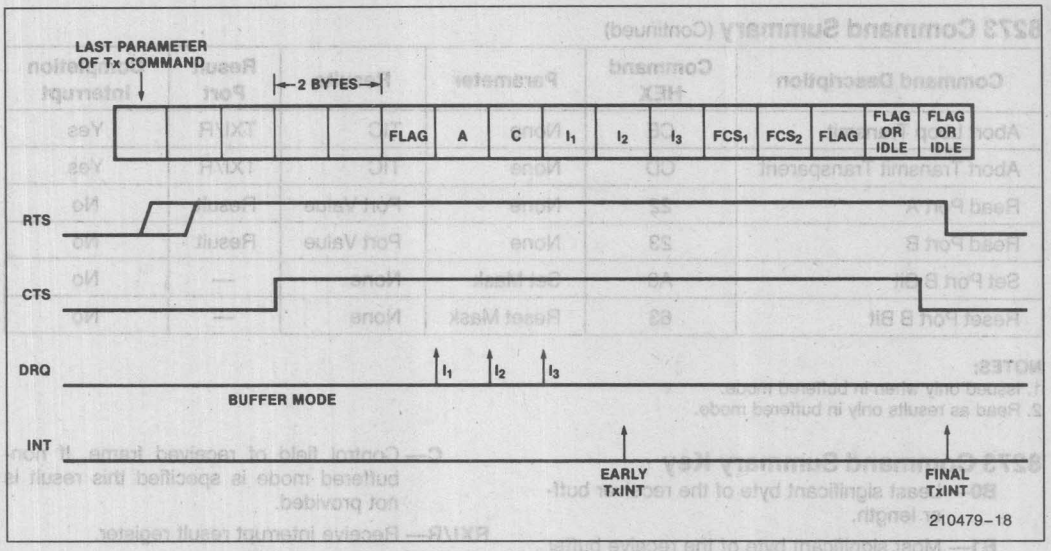


Figure 16a. Typical Frame Transmission, Buffered Mode

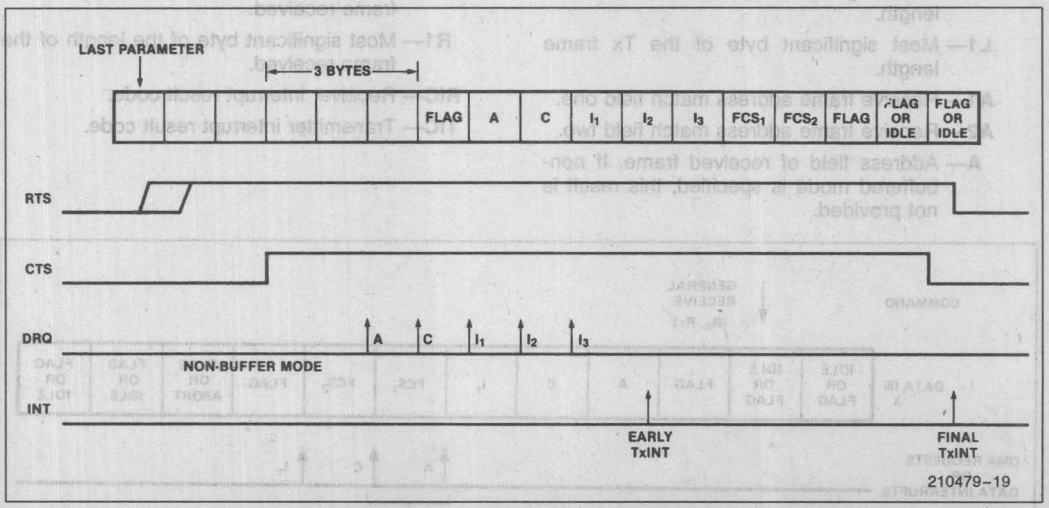


Figure 16b. Typical Frame Transmission, Non-Buffered Mode

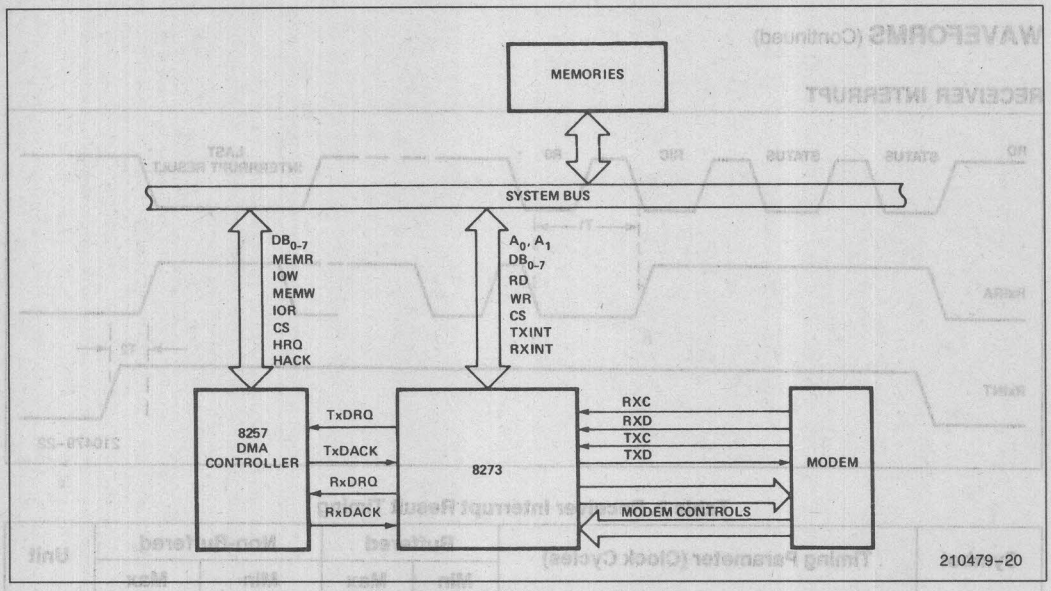


Figure 17. 8273 System Diagram

WAVEFORMS

COMMAND PHASE

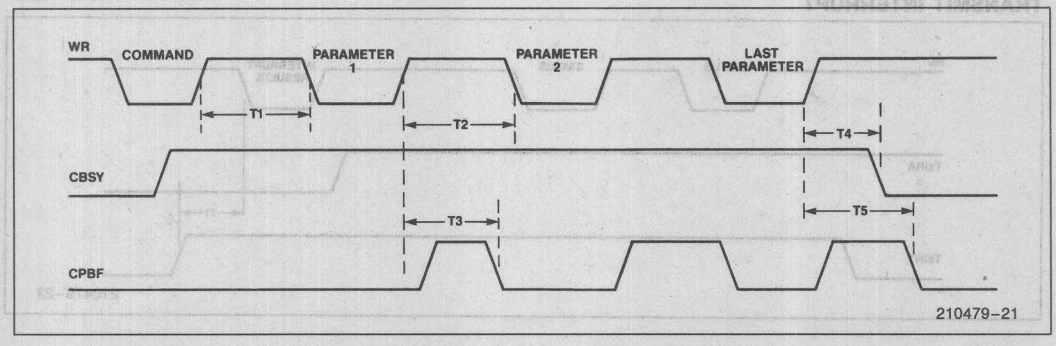


Table 2. Command Phase Timing (Full Duplex)

Symbol	Timing Parameter	Buffered		Non-Buffered		Unit
		Min	Max	Min	Max	
T1	Between Command & First Parameter	13	756	13	857	tcy
T2	Between Consecutive Parameters	10	604	10	705	tcy
T3	Command Parameter Buffer Full Bit Reset after Parmeter Loaded	10	604	10	705	tcy
T4	Command Busy Bit Reset after Last Parameter	128	702	128	803	tcy
T5	CPBF Bit Reset after Last Parameter	10	604	10	705	tcy

WAVEFORMS (Continued)

RECEIVER INTERRUPT

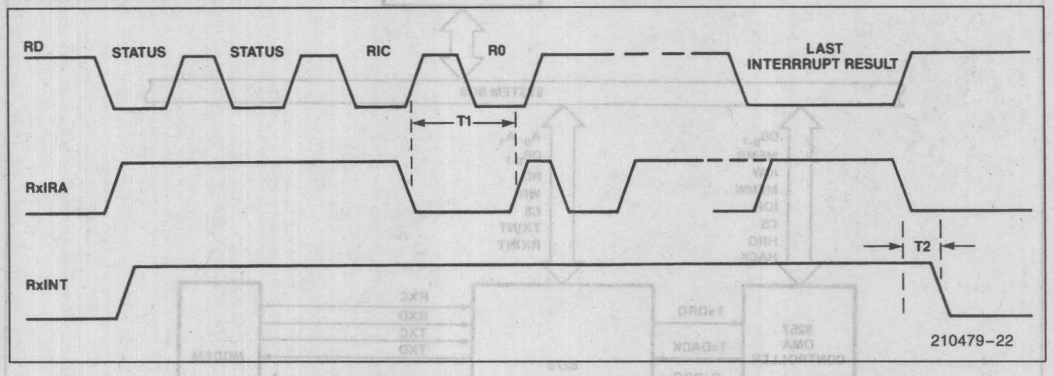


Table 3. Receiver Interrupt Result Timing

Symbol	Timing Parameter (Clock Cycles)	Buffered		Non-Buffered		Unit
		Min	Max	Min	Max	
T1	RxIRA Bit Set after RIC Read	18	29	18	29	tcy
T2	RxINT Goes Away after Last Int. Result Read	16	27	16	27	tcy

TRANSMIT INTERRUPT

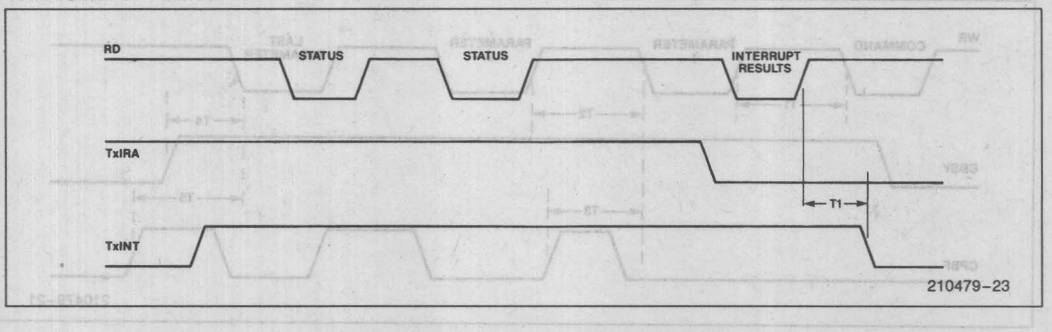


Table 4. Transmit Interrupt Result

Symbol	Timing (Clock Cycle)	Buffered		Non-Buffered		Unit
		Min	Max	Min	Max	
T1	TxINT Inactive after Int. Results Read	13	353	13	454	tcy

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin With Respect to Ground.....−0.5V to +7V
Power Dissipation .....1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS 8273 (T_A = 0°C to 70°C, V_{CC} = +5.0V ±5%)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2.0 mA for Data Bus Pins I_{OL} = 1.0 mA for Output Port Pins I_{OL} = 1.6 mA for All Other Pins
V_{OH}	Output High Voltage	2.4		V	I_{OH} = − 200 μA for Data Bus Pins I_{OH} = −100 μA for All Other Pins
I_{IL}	Input Load Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Leakage Current		±10	μA	V_{OUT} = V_{CC} to 0.45V
I_{CC}	V_{CC} Supply Current		180	mA	

CAPACITANCE 8273 (T_A = 25°C, V_{CC} = GND = 0V)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	t_c = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

A.C. CHARACTERISTICS (T_A = 0°C to 70°C, V_{CC} = +5.0V ±5%)

CLOCK TIMING (8273)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{CY}	Clock	250		1000	ns	64K Baud Max Operating Rate
t_{CL}	Clock Low	120			ns	
t_{CH}	Clock High	120			ns	

A.C. CHARACTERISTICS 8273 (T_A = 0°C to 70°C, V_{CC} = +5.0V ± 5%)

READ CYCLE

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{RD}$	0		ns	(Note 2)
t_{CA}	Select Hold from $\overline{RD}$	0		ns	(Note 2)
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	Data Delay from Address		300	ns	(Note 2)
t_{RD}	Data Delay from $\overline{RD}$		200	ns	C_L = 150 pF, (Note 2)
t_{DF}	Output Float Delay	20	100	ns	C_L = 20 pF For Minimum; 150 pF for Maximum
t_{DC}	DACK Setup to $\overline{RD}$	25		ns	
t_{CD}	DACK Hold from $\overline{RD}$	25		ns	
t_{KD}	Data Delay from DACK		300	ns	

WRITE CYCLE

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AC}	Select Setup to $\overline{WR}$	0		ns	
t_{CA}	Select Hold from $\overline{WR}$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$	150		ns	
t_{WD}	Data Hold from $\overline{WR}$	0		ns	
t_{DC}	DACK Setup to $\overline{WR}$	25		ns	
t_{CD}	DACK Hold from $\overline{WR}$	25		ns	

DMA

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{CQ}	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		200	ns	

OTHER TIMING

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{RSTW}	Reset Pulse Width	10		t_{CY}	
t_r	Input Signal Rise Time		20	ns	
t_f	Input Signal Fall Time		20	ns	
t_{RSTS}	Reset to First $\overline{IOWR}$	2		t_{CY}	
t_{CY32}	32X Clock Cycle Time	$13.02 \times t_{CY}$		ns	
t_{CL32}	32X Clock Low Time	$4 \times t_{CY}$		ns	
t_{CH32}	32X Clock High Time	$4 \times t_{CY}$		ns	
t_{DPLL}	$\overline{DPLL}$ Output Low	$1 \times t_{CY} - 50$		ns	

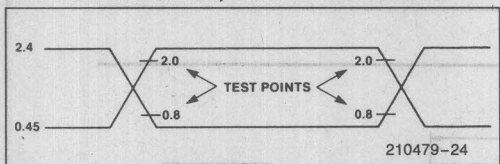
A.C. CHARACTERISTICS 8273 (T_A = 0°C to 70°C, V_{CC} = +5.0V ± 5%) (Continued)

OTHER TIMING (Continued)

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{DCL}	Data Clock Low	$1 \times t_{CY} - 50$		ns	
t_{DCH}	Data Clock High	$2 \times t_{CY}$		ns	
t_{DCY}	Data Clock	$62.5 \times t_{CY}$		ns	(Note 3)
t_{TD}	Transmit Data Delay		200	ns	
t_{DS}	Data Setup Time	200		ns	
t_{DH}	Data Hold Time	100		ns	
t_{FLD}	$\overline{\text{FLAG DET}}$ Output Low	$8 \times t_{CY} \pm 50$		ns	

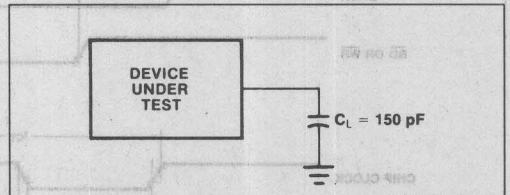
NOTES:
1. All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V; Output "1" at 2.0V, "0" at 0.8V.
2. t_{AD}, t_{RD}, t_{AC}, and t_{CA} are not concurrent specs.
3. If receive commands or Read/Write Port commands are issued while both the transmitter and receiver are active, this specification will be 81.5 T_{CY} min.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a logic "1" and 0.45V for a logic "0". Timing measurements are made at 2.0V for a logic "1" and 0.8V for a logic "0".

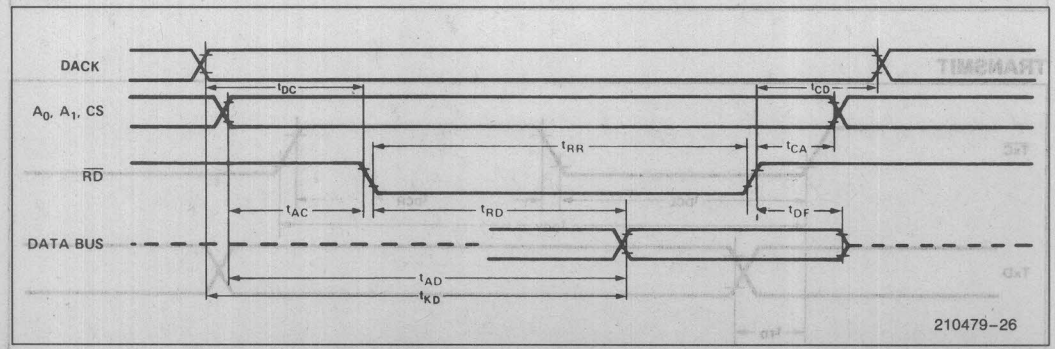
A.C. TESTING LOAD CIRCUIT



C_L = 150 pF
C_L Includes Jig Capacitance

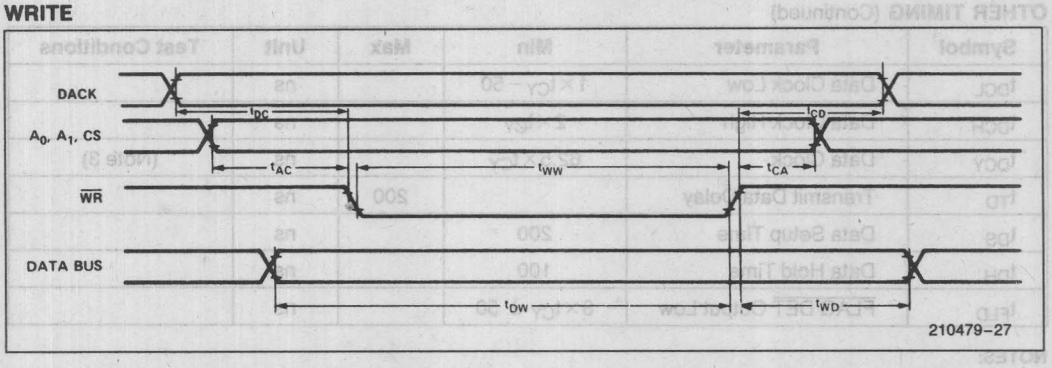
WAVEFORMS

READ

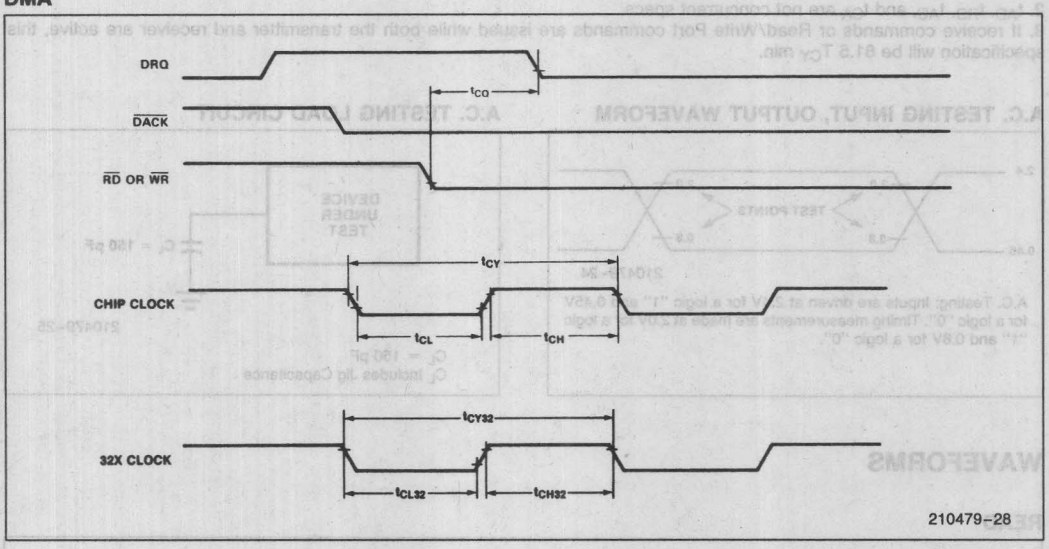


WAVEFORMS (Continued)

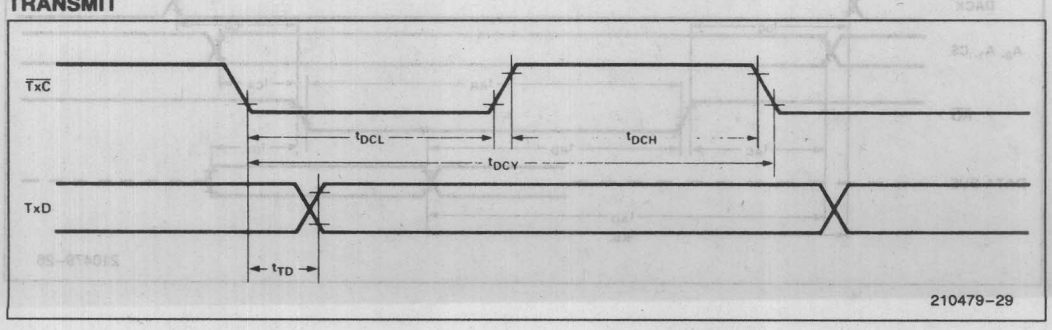
WRITE



DMA

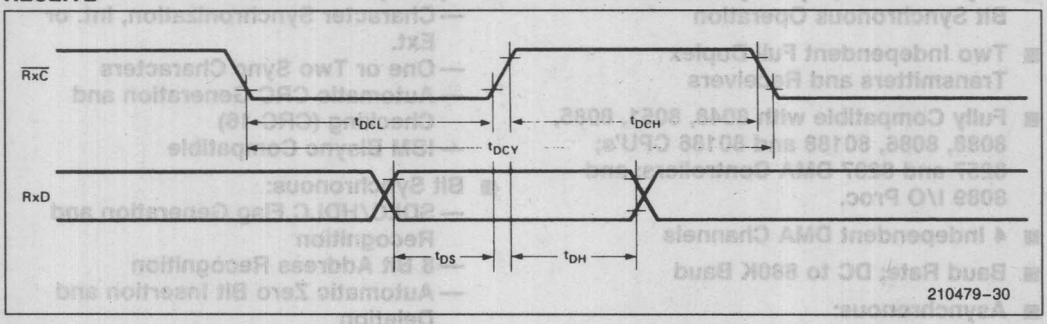


TRANSMIT

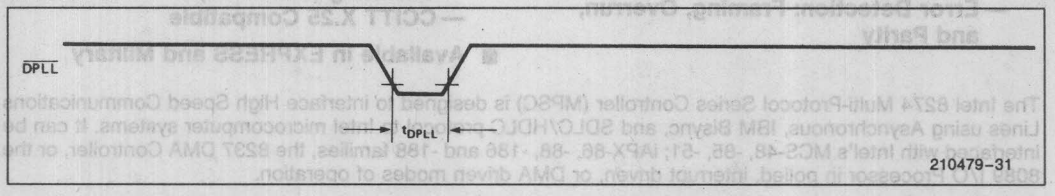


WAVEFORMS (Continued)

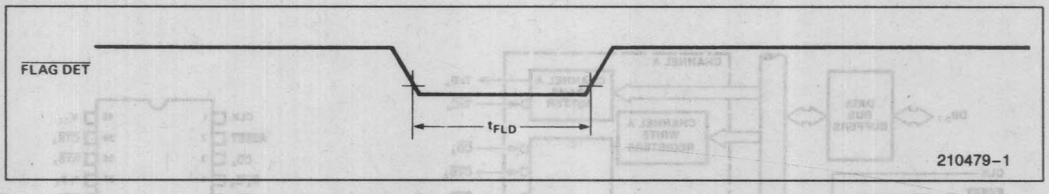
RECEIVE



DPLL OUTPUT



FLAG DETECT OUTPUT

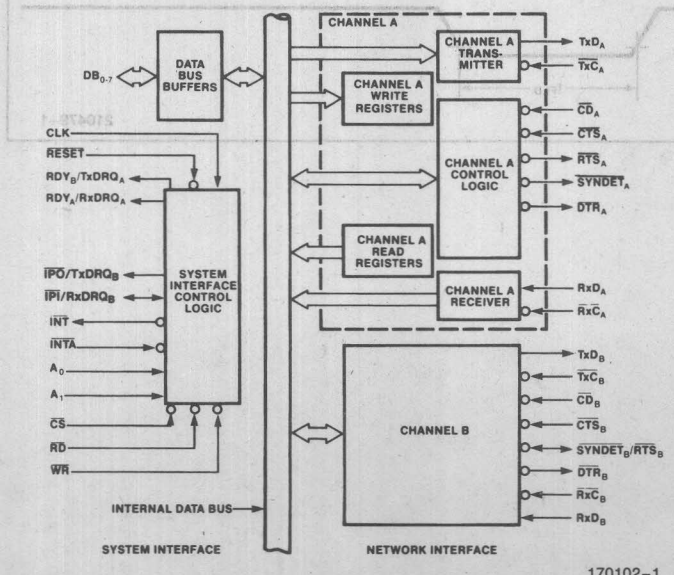


8274
MULTI-PROTOCOL SERIAL CONTROLLER (MPSC)

- **Asynchronous, Byte Synchronous and Bit Synchronous Operation**
- **Two Independent Full Duplex Transmitters and Receivers**
- **Fully Compatible with 8048, 8051, 8085, 8088, 8086, 80188 and 80186 CPU's; 8257 and 8237 DMA Controllers; and 8089 I/O Proc.**
- **4 Independent DMA Channels**
- **Baud Rate: DC to 880K Baud**
- **Asynchronous:**
  - **5–8 Bit Character; Odd, Even, or No Parity; 1, 1.5 or 2 Stop Bits**
  - **Error Detection: Framing, Overrun, and Parity**
- **Byte Synchronous:**
  - **Character Synchronization, Int. or Ext.**
  - **One or Two Sync Characters**
  - **Automatic CRC Generation and Checking (CRC-16)**
  - **IBM Bisync Compatible**
- **Bit Synchronous:**
  - **SDLC/HDLC Flag Generation and Recognition**
  - **8 Bit Address Recognition**
  - **Automatic Zero Bit Insertion and Deletion**
  - **Automatic CRC Generation and Checking (CCITT-16)**
  - **CCITT X.25 Compatible**
- **Available in EXPRESS and Military**

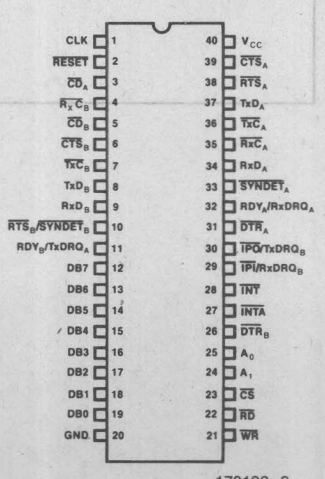
The Intel 8274 Multi-Protocol Series Controller (MPSC) is designed to interface High Speed Communications Lines using Asynchronous, IBM Bisync, and SDLC/HDLC protocol to Intel microcomputer systems. It can be interfaced with Intel's MCS-48, -85, -51; iAPX-86, -88, -186 and -188 families, the 8237 DMA Controller, or the 8089 I/O Processor in polled, interrupt driven, or DMA driven modes of operation.

The MPSC is a 40 pin device fabricated using Intel's High Performance HMOS Technology.



170102-1

Figure 1. Block Diagram



170102-2

Figure 2. Pin Configuration

November 1986
Order Number: 170102-002

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
CLK	1	I	**CLOCK:** System clock, TTL compatible.
$\overline{RESET}$	2	I	**RESET:** A low signal on this pin will force the MPSC to an idle state. TxD_A and TxD_B are forced high. The modem interface output signals are forced high. The MPSC will remain idle until the control registers are initialized. Reset must be true for one complete CLK cycle.
$\overline{CD}_A$	3	I	**CARRIER DETECT (CHANNEL A):** This interface signal is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD_A line. If the auto enable control is set the 8274 will not enable the serial receiver until $\overline{CD}_A$ has been activated.
$\overline{RxC}_B$	4	I	**RECEIVE CLOCK (CHANNEL B):** The serial data are shifted into the Receive Data input (RxD_B) on the rising edge of the Receive Clock.
$\overline{CD}_B$	5	I	**CARRIER DETECT (CHANNEL B):** This interface signal is supplied by the modem to indicated that a data carrier signal has been detected and that a valid data signal is present on the RxD_B line. If the auto enable control is set the 8274 will not enable the serial receiver until $\overline{CD}_B$ has been activated.
$\overline{CTS}_B$	6	I	**CLEAR TO SEND (CHANNEL B):** This interface signal is supplied by the modem in response to an active $\overline{RTS}$ signal. $\overline{CTS}$ indicates that the data terminal/computer equipment is permitted to transmit data. In addition, if the auto enable control is set, the 8274 will not transmit data bytes until $\overline{CTS}$ has been activated.
$\overline{TxC}_B$	7	I	**TRANSMIT CLOCK (CHANNEL B):** The serial data are shifted out from the Transmit Data output (TxD_B) on the falling edge of the Transmit Clock.
TxD_B	8	O	**TRANSMIT DATA (CHANNEL B):** This pin transmits serial data to the communications channel (Channel B).
RxD_B	9	I	**RECEIVE DATA (CHANNEL B):** This pin receives serial data from the communications channel (Channel B).
$\overline{SYNDET}_B$ /$\overline{RTS}_B$	10	I/O	**SYNCHRONOUS DETECTION (CHANNEL B):** This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating Flag detection. In asynchronous mode it is a general purpose input (Channel B). **REQUEST TO SEND (CHANNEL B):** General purpose output, generally used to signal that Channel B is ready to send data. When the RTS bit is reset in asynchronous mode, the signal does not go inactive (High) until the transmitter is empty. $\overline{SYNDET}_B$ or $\overline{RTS}_B$ selection is done by WR2; D7. (Channel A).

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
RDY_B/ $TxDRQ_A$	11	O	**READY (CHANNEL B)/TRANSMITTER DMA REQUEST (CHANNEL A):** In mode 0 this pin is RDY_B and is used to synchronize data transfers between the processor and the MPSC (Channel B). In modes 1 and 2 this pin is $TxDRQ_A$ and is used by the Channel A transmitter to request a DMA transfer.
DB7	12	I/O	**DATA BUS:** The Data Bus lines are bidirectional three state lines which interface with the system's Data Bus.
DB6	13		
DB5	14		
DB4	15		
DB3	16		
DB2	17		
DB1	18		
DB0	19		
GND	20		**GROUND.**
V_{CC}	40		**POWER:** +5V Supply
$\overline{CTS}_A$	39	I	**CLEAR TO SEND (CHANNEL A):** This interface signal is supplied by the Modem in response to an active $\overline{RTS}$ signal. $\overline{CTS}$ indicates that the data terminal/computer equipment is permitted to transmit data. In addition, if the auto enable control is set, the 8274 will not transmit data bytes until $\overline{CTS}$ has been activated.
$\overline{RTS}_A$	38	O	**REQUEST TO SEND (CHANNEL A):** General purpose output commonly used to signal that Channel A is ready to send data. When the RTS bit is reset in asynchronous mode, the signal does not go inactive (High) until the transmitter is empty.
TxD_A	37	O	**TRANSMIT DATA (CHANNEL A):** This pin transmits serial data to the commmunications channel (Channel A).
$\overline{TxC}_A$	36	I	**TRANSMIT CLOCK (CHANNEL A):** The serial data are shifted out from the Transmit Data output (TxD_A) on the falling edge of the Transmit Clock.
$\overline{RxC}_A$	35	I	**RECEIVE CLOCK (CHANNEL A):** The serial data are shifted into the Receive Data input (RxD_A) on the rising edge of the Receive Clock.
RxD_A	34	I	**RECEIVE DATA (CHANNEL A):** This pin receives serial data from the communications channel (Channel A).
$\overline{SYNDET}_A$	33	I/O	**SYNCHRONOUS DETECTION (CHANNEL A):** This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating flag detection. In asynchronous mode it is a general purpose input (Channel A).

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
RDY_A/ $RxDRQ_A$	32	O	**READY:** In mode 0 this pin is RDY_A and is used to synchronize data transfers between the processor and the MPSC (Channel A). In modes 1 and 2 this pin is $RxDRQ_A$ and is used by the channel A receiver to request a DMA transfer.
$\overline{DTR}_A$	31	O	**DATA TERMINAL READY (CHANNEL A):** General purpose output.
$\overline{IPO}$/ $TxDRQ_B$	30	O	**INTERRUPT PRIORITY OUT/TRANSMITTER DMA REQUEST (CHANNEL B):** In modes 0 and 1, this pin is Interrupt Priority Out. It is used to establish a hardware interrupt priority scheme with $\overline{IPI}$. It is low only if $\overline{IPI}$ is low and the controlling processor is not servicing an interrupt from this MPSC. In mode 2 it is $TxDRQ_B$ and is used to request a DMA cycle for a transmit operation (Channel B).
$\overline{IPI}$/ $RxDRQ_B$	29	I/O	**INTERRUPT PRIORITY IN/RECEIVER DMA REQUEST (CHANNEL B):** In modes 0 and 1, $\overline{IPI}$ is Interrupt Priority In. A low on $\overline{IPI}$ means that no higher priority device is being serviced by the controlling processor's interrupt service routine. In mode 2 this pin is $RxDRQ_B$ and is used to request a DMA cycle for a receive operation (Channel B).
$\overline{INT}$	28	O	**INTERRUPT:** The interrupt signal indicates that the highest priority internal interrupt requires service (open collector). Priority can be resolved via an external interrupt controller or a daisy-chain scheme.
$\overline{INTA}$	27	I	**INTERRUPT ACKNOWLEDGE:** This Interrupt Acknowledge signal allows the highest priority interrupting device to generate an interrupt vector. This pin must be pulled high (inactive) in non-vector mode.
$\overline{DTR}_B$	26	O	**DATA TERMINAL READY (CHANNEL B):** This is a general purpose output.
A_0	25	I	**ADDRESS:** This line selects Channel A or B during data or command transfers. A low selects Channel A.
A_1	24	I	**ADDRESS:** This line selects between data or command information transfer. A low means data.
$\overline{CS}$	23	I	**CHIP SELECT:** This signal selects the MSPC and enables reading from or writing into registers.
$\overline{RD}$	22	I	**READ:** Read controls a data byte or status byte transfer from the MPSC to the CPU.
$\overline{WR}$	21	I	**WRITE:** Write controls transfer of data or commands to the MPSC.

Bisync), and Bit Synchronous (ISO's HDLC, IBM's SDLC) protocols. This controller's flexible architecture allows easy implementation of many variations of these three protocols with low software and hardware overhead.

The Multi-Protocol Serial controller (MPSC) implements two independent serial receiver/transmitter channels.

The MPSC supports several microprocessor interface options: Polled, Wait, Interrupt driven and DMA driven. The MPSC is designed to support INTEL's MCS-85 and iAPX 86, 88, 186, 188 families.

FUNCTIONAL DESCRIPTION

Additional information on Asynchronous and Synchronous Communications with the 8274 is available respectively in the Applications Notes AP 134 and AP 145.

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers for each channel, 2 readable registers for Channel A and 3 readable registers for Channel B).

In the following discussion, the writable registers will be referred to as WRO through WR7 and the readable registers will be referred to as RRO through RR2.

This section of the data sheet describes how the Asynchronous and Synchronous protocols are implemented in the MPSC. It describes general considerations, transmit operation, and receive operation for Asynchronous, Byte Synchronous, and Bit Synchronous protocols.

bit rate. If the X1 clock mode is selected, the bit synchronization must be accomplished externally.

2. Number of Stop Bits. This parameter is specified by bits 2 and 3 of WR4. The number of stop bits may be set to 1, 1½, or 2.
3. Parity Selection. Parity may be set for odd, even, or no parity by bits 0 and 1 of WR4.
4. Receiver Character Length. This parameter sets the length of received characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 6 and 7 of WR3.
5. Receiver Enable. The serial-channel receiver operation may be enabled or disabled by setting or clearing bit 0 of WR3.
6. Transmitter Character Length. This parameter sets the length of transmitted characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 5 and 6 of WR5. Characters of less than 5 bits in length may be transmitted by setting the transmitted length to five bits (set bits 5 and 6 of WR5 to 0).

   The MPSC then determines the actual number of bits to be transmitted from the character data byte. The bits to be transmitted must be right justified in the data byte, the next three bits must be set to 0 and all remaining bits must be set to 1. The following table illustrates the data formats for transmission of 1 to 5 bits of data.

Byte Written								Number of Bits Transmitted
D7	D6	D5	D4	D3	D2	D1	D0	(Character Length)
1	1	1	1	0	0	0	c	1
1	1	1	0	0	0	c	c	2
1	1	0	0	0	c	c	c	3
1	0	0	0	c	c	c	c	4
0	0	0	c	c	c	c	c	5

7. Transmitter Enable. The serial channel transmitter operation may be enabled or disabled by setting or clearing bit 3 of WR5.
8. Interrupt Mode. Specified by bits 3 and 4 of WR1.

For data transmission via a modem or RS-232-C interface, the following information must also be specified:

1. The Request To Send (RTS) (WR5; D1) and Data Terminal Ready (DTR) (WR5; D7) bits must be set along with the Transmit Enable bit (WR5; D3).
2. Auto Enable may be set to allow the MPSC to automatically enable the channel transmitter when the clear-to-send signal is active and to automatically enable the receiver when the carrier-detect signal is active. However, the Transmit Enable bit (WR3; D3) and Receive Enable bit (WR3; D1) must be set in order to use the Auto Enable mode. Auto Enable is controlled by bit 5 of WR3.

When loading Initialization parameters into the MPSC, WR4 information must be written before the WR1, WR3, WR5 parameters commands.

During initialization, it is desirable to guarantee that the external/status latches reflect the latest interface information. Since up to two state changes are internally stored by the MPSC, at least two Reset External/Status Interrupt commands must be issued. This procedure is most easily accomplished by simply issuing this reset command whenever the pointer register is set during initialization.

An MPSC initialization procedure (MPSCRXINIT) for asynchronous communication is listed in Intel Application Note AP 134.

TRANSMIT

The transmit function begins when the Transmit Enable bit (WR5; D3) is set. The MPSC automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits (1, 1.5 or 2 bits) to the data character being transmitted. 1.5 stop bits option must be used with X16, X32 or X64 clock options only. The data character is transmitted least significant bit first.

The serial data are shifted out from the Transmit Data (TxD) output on the falling edge of the Transmit Clock (TxC) input at a rate programmable to 1, $1/16$th, $1/32$nd, or $1/64$th of the clock rate supplied to the TxC input.

The TxD output is held high when the transmitter has no data to send, unless, under program control, the Send Break (WR5; D4) command is issued to hold the TxD low.

If the External/Status Interrupt bit (WR1; D0) is set, the status of $\overline{CD}$, $\overline{CTS}$ and $\overline{SYNDET}$ are monitored and, if any changes occur for a period of time greater than the minimum specified pulse width, an interrupt is generated. $\overline{CTS}$ is usually monitored using this interrupt feature (e.g., Auto Enable option).

The Transmit Buffer Empty bit (RRO; D2) is set by the MPSC when the data byte from the buffer is loaded in the transmit shift register. Data should be written to the MPSC only when the Tx buffer becomes empty to prevent overwriting.

Receive

The receive function begins when the Receive Enable (WR3; D0) bit is set. If the Auto Enable (WR3; D5) option is selected, then Carrier Detect ($\overline{CD}$) must also be low. A valid start bit is detected if a low persists for at least ½ bit time on the Receive Data (RxD) input.

The data is sampled at mid-bit time, on the rising edge of RxC, until the entire character is assembled. The receiver inserts 1's when a character is less than 8 bits. If parity (WR4; D0) is enabled and the character is less than 8 bits the parity bit is not stripped from the character.

Error Reporting

The receiver also stores error status for each of the 3 data characters in the data buffer. Three error conditions may be encountered during data reception in the asynchronous mode:

1. **Parity.** If parity bits are computed and transmitted with each character and the MPSC is set to check parity (bit 0 in WR4 is set), a parity error will occur whenever the number of "1" bits within the character (including the parity bit) does not match the odd/even setting of the parity check flag (bit 1 in WR4). When a parity error is detected, the parity error flag (RR1; D4) is set and remains set until it is reset by the Error Reset command (WR0; D5, D4, D3).
2. **Framing.** A framing error will occur if a stop bit is not detected immediately following the parity bit (if parity checking is enabled) or immediately following the most-significant data bit (if parity checking is not enabled). When a Framing Error is detected, the Framing Error bit (RR1; D6) is set and remains set until reset by the Error Reset Command (WR0; D5, D4, D3). The detection of a Framing Error adds an additional ½ bit time to the character time so the Framing Error is not interpreted as a new start bit.
3. **Overrun.** If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1; D5) is set. When this occurs, the fourth character assembled replaces the third character in the receive buffers. Only the overwritten character is flagged with the Receive Overrun bit. The Receive Overrun bit (RR1; D5) is reset by the Error Reset command (WR0; D5, D4, D3).

External/Status Latches

The MPSC continuously monitors the state of five external/status conditions:

1. CTS—clear-to-send input pin.
2. CD—carrier-detect input pin.
3. SYNDET—sync-detect input pin. This pin may be used as a general-purpose input in the asynchronous communication mode.
4. BREAK—a break condition (series of space bits on the receiver input pin).
5. TxUNDERRUN/EOM—Transmitter Underrun/End of Message.

A change of state in any of these monitored conditions will cause the associated status bit in RR0 to be latched (and optionally cause an interrupt).

If the External/Status Interrupt bit (WR1; D0) is enabled, Break Detect (RR0; D7) and Carrier Detect (RR0; D3) will cause an interrupt. Reset External/Status interrupts (WR0; D5, D4, D3) will clear Break Detect and Carrier Detect bits if they are set.

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers for each channel, 2 readable registers for Channel A and 3 readable registers for Channel B). They are all accessed via the command ports.

An internal pointer register selects which of the command or status registers will be read or written during a commmand/status access of an MPSC channel.

After reset, the contents of the pointer registers are zero. The first write to a command register causes the data to be loaded into Write Register 0 (WR0). The three least significant bits of WR0 are loaded into the Command/Status Pointer. The next read or write operation accesses the read or write register selected by the pointer. The pointer is reset after the read or write operation is completed.

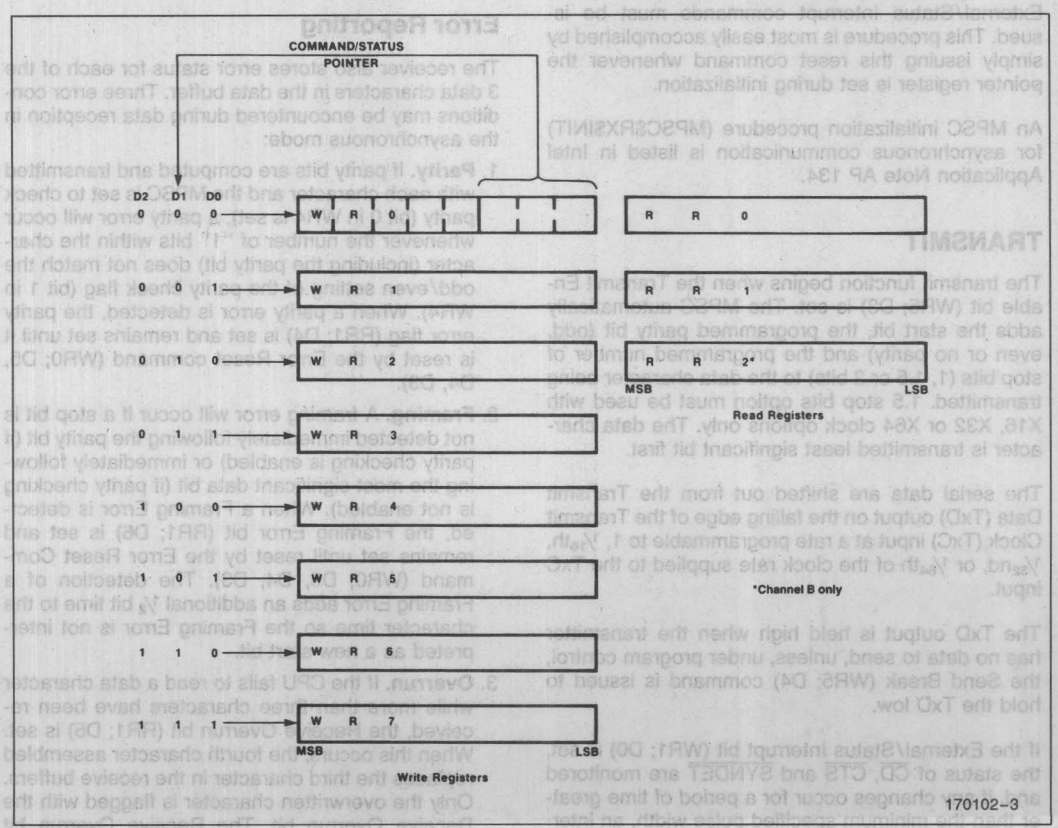


Figure 3. Command/Status Register Architecture (each serial channel)

Asynchronous Mode Register Setup

	D7	D6	D5	D4	D3	D2	D1	D0
WR3	00 Rx 5 b/char 01 Rx 7 b/char 10 Tx 6 b/char 11 Rx 8 b/char		AUTO ENABLE	0	0	0	0	Rx ENABLE
WR4	00 X1 Clock 01 X16 Clock 10 X32 Clock 11 X64 Clock		0	0	01 1 STOP BIT 10 1½ STOP BITS 11 2 STOP BITS		EVEN/ $\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00 Tx ≤ 5 b/char 01 Tx 7 b/char 10 Tx 6 b/char 11 Tx 8 b/char		SEND BREAK	Tx ENABLE	0	RTS	0

SYNCHRONOUS OPERATION—MONOSYNC, BISYNC

General

The MPSC must be initialized with the following parameters: odd or even parity (WR4; D1, D0), X1 clock mode (WR4; D7, D6), 8- or 16-bit sync character (WR4; D5, D4), CRC polynomial (WR5; D2), Transmitter Enable (WR5; D3), interrupt modes (WR1, WR2), transmit character length (WR5; D6, D5) and receive character length (WR3; D7, D6). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The data is transmitted on the falling edge of the Transmit Clock, ($\overline{\text{TxC}}$) and is received on the rising edge of Receive Clock ($\overline{\text{RxC}}$). The X1 clock is used for both transmit and receive operations for all three sync modes: Mono, Bi and External.

Transmit Set-Up—Monosync, Bisync

Transmit data is held high after channel reset, or if the transmitter is not enabled. A break may be programmed to generate a spacing line that begins as soon as the Send Break (WR5; D4) bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

Using interrupts for data transfer requires that the Transmit Interrupt/DMA Enable bit (WR1; D1) be set. An interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied either by writing another character into the transmitter or by resetting the Transmitter Interrupt/DMA Pending latch with a Reset Transmitter Inter-

Synchronous Mode Register Setup—Monosync, Bisync

	D7	D6	D5	D4	D3	D2	D1	D0
WR3	00 Rx 5 b/char 01 Rx 7 b/char 10 Tx 6 b/char 11 Rx 8 b/char		AUTO ENABLE	ENTER HUNT MODE	Rx CRC ENABLE	0	SYNC CHAR LOAD INHIBIT	Rx ENABLE
WR4	0	0	00 8 bit Sync 01 16 bit Sync 11 Ext Sync		0	0	EVEN/ $\overline{\text{ODD}}$ PARITY	PARITY ENABLE
WR5	DTR	00 Tx ≤ 5 b/char 01 Tx 7 b/char 10 Tx 6 b/char 11 Tx 8 b/char		SEND BREAK	Tx ENABLE	1 (SELECTS CRC-16)	RTS	Tx CRC ENABLE

rupt/DMA Pending Command (WR0; D5, D4, D3). If nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupt, but this situation does cause a Transmit Underrun condition (RR0; D6).

Data Transfers using the RDY signal are for software controlled data transfers such as block moves. RDY tells the CPU that the MPSC is not ready to accept/provide data and that the CPU must extend the output/input cycle. DMA data transfers use the TxDRQ A/B signals which indicate that the transmit buffer is empty, and that the MPSC is ready to accept the next data character. If the data character is not loaded into the MPSC by the time the transmit shift register is empty, the MPSC enters the Transmit Underrun condition.

The MPSC has two programmable options for solving the transmit underrun condition: it can insert sync characters, or it can send the CRC characters generated so far, followed by sync characters. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RR0; D6) is in a set condition allowing the insertion of sync characters when there is no data to send. The CRC is not calculated on these automatically inserted sync characters. When the CPU detects the end message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data to send.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded in the Transmit Shift Register. The status register indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded into the Tx Shift Register). If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter (WR5; D3).

Bisync CRC Generation. Setting the Transmit CRC enable bit (WR5; D0) indicates CRC accumulation when the program sends the first data character to the MPSC. Although the MPSC automatically transmits up to two sync characters (16 bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The Transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded into the transmit shift register. To ensure this bit in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the MPSC.

Transmit Transparent Mode. Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16 bit sync characters. Exclusion of DLE characters from CRC calculation can be achieved by disabling CRC calculation immediately preceding the DLE character transfer to the MPSC.

In the transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16) (WR4; D5, D4). When in the Monosync mode, the transmitter sends from WR6 and the receiver compares against WR7. One or two CRC polynomials, CRC 16 or SDLC, may be used with synchronous modes. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command (WR0; D7, D6).

The External/Status interrupt (WR1; D0) mode can be used to monitor the status of the $\overline{CTS}$ input as well as the Transmit Underrun/EOM latch. Optionally, the Auto Enable (WR3; D5) feature can be used to enable the transmitter when $\overline{CTS}$ is active. The first data transfer to the MPSC can begin when the External/Status interrupt (CTS (RR0; D5) status bit set) occurs following the Transmit Enable command (WR5; D3).

Receive

After a channel reset, the receiver is in the Hunt phase, during which the MPSC looks for character synchronization. The Hunt begins only when the receiver is enabled and data transfer begins only when character synchronization has been achieved. If character synchronization is lost, the hunt phase can be re-entered by writing the Enter Hunt Phase (WR3; D4) bit. The assembly of received data continues until the MPSC is reset or until the receiver is disabled (by command or by $\overline{CD}$ while in the Auto Enables mode) or until the CPU sets the Enter Hunt Phase bit. Under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit (WR3; D1) bit. After character synchronization is achieved the assembled characters are transferred to the receive data FIFO. After

receiving the first data character, the Sync Character Load Inhibit bit should be reset to zero so that all characters are received, including the sync characters. This is important because the received CRC may look like a sync character and not get received.

Data may be transferred with or without interrupts. Transferring data without interrupts is used for a purely polled operation or for off-line conditions. There are two interrupt modes available for data transfer: Interrupt on First Character Only and Interrupt on Every Character.

Interrupt on First Character Only mode is normally used to start a polling loop, a block transfer sequence using RDY to synchronize the CPU to the incoming data rate, or a DMA transfer using the RxDRQ signal. The MPSC interrupts on the first character and thereafter only interrupts after a Special Receive Condition is detected. This mode can be reinitialized using the Enable Interrupt On Next Receive Character (WR0; D5, D4, D3) command which allows the next character received to generate an interrupt. Parity Errors do not cause interrupts, but End of Frame (SDLC operation) and Receive Overrun do cause interrupts in this mode. If the external status interrupts (WR1; D0) are enabled an interrupt may be generated any time the $\overline{CD}$ changes state.

Interrupt On Every Character mode generates an interrupt whenever a character enters the receive buffer. Errors and Special Receive Conditions generate a special vector if the Status Affects Vector (WR1B; D2) is selected. Also the Parity Error may be programmed (WR1; D4, D3) not to generate the special vector while in the Interrupt On Every Character mode.

The Special Receive Condition interrupt can only occur while in the Receive Interrupt On First Character Only or the Interrupt On Every Receive Character modes. The Special Receive Condition interrupt is caused by the Receive Overrun (RR1; D5) error condition. The error status reflects an error in the current word in the receive buffer, in addition to any Parity or Overrun errors since the last Error Reset (WR0; D5, D4, D3). The Receive Overrun and Parity error status bits are latched and can only be reset by the Error Reset (WR0; D5, D4, D3) command.

The CRC check result may be obtained by checking for CRC bit (RR1; D6). This bit gives the valid CRC result 16 bit times after the second CRC byte has been read from the MPSC. After reading the second CRC byte, the user software must read two more characters (may be sync characters) before checking for CRC result in RR1. Also for proper CRC computation by the receiver, the user software must reset the Receive CRC Checker (WR0; D7, D6) after receiving the first valid data character. The receive CRC Enable bit (WR3; D3) may also be enabled at this time.

SYNCHRONOUS OPERATION—SDLC

General

Like the other synchronous operations the SDLC mode must be initialized with the following parameters: SDLC mode (WR4; D5, D4), SDLC polynomial (WR5; D2), Request to Send, Data Terminal Ready, transmit character length (WR5; D6, D5), interrupt modes (WR1; WR2), Transmit Enable (WR5; D3), Receive Enable (WR3; D0), Auto Enable (WR3; D5) and External/Status Interrupt (WR1; D0). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The Interrupt modes for SDLC operation are similar to those discussed previously in the synchronous operations section.

Synchronous Mode Register Setup—SDLC/HDLC

	D7	D6	D5	D4	D3	D2	D1	D0
WR3	00 Rx 5 b/char 01 Rx 7 b/char 10 Rx 6 b/char 11 Rx 8 b/char		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	ADDRESS SEARCH MODE	0	Rx
WR4	0	0	1	0 (SELECTS SDLC/ HDLC MODE)	0	0	0	0
WR5	DTR	00 Tx ≤ 5 b/char 01 Tx 7 b/char 10 Tx 6 b/char 11 Tx 8 b/char		SEND BREAK	Tx ENABLE	0 (SELECTS SDLC/HDLC CRC)	RTS	Tx CRC ENABLE

Transmit

After a channel reset, the MPSC begins sending SDLC flags.

Following the flags in an SDLC operation the 8-bit address field, control field and information field may be sent to the MPSC by the microprocessor. The MPSC transmits the Frame Check Sequence using the Transmit Underrun feature. The MPSC automatically inserts a zero after every sequence of 5 consecutive 1's except when transmitting Flags or Aborts.

SDLC—like protocols do not have provision for fill characters within a message. The MPSC therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by sending the two bytes of CRC and then one or more flags. This allows very high-speed transmissions under DMA or CPU control without requiring the CPU to respond quickly to the end-of-message situation.

After a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Flag characters are sent. The MPSC begins to send the frame when data is written into the transmit buffer. Between the time the first data byte is written, and the end of the message, the Reset Transmit Underrun/EOM (WR0; D7, D6) command must be issued. The Transmit Underrun/EOM status bit (RR0; D6) is in the reset state at the end of the message which automatically sends the CRC characters.

The MPSC may be programmed to issue a Send Abort command (WR0; D5, D4, D3). This command causes at least eight 1's but less than fourteen 1's to be sent before the line reverts to continuous flags.

Receive

After initialization, the MPSC enters the Hunt phase, and remains in the Hunt phase until the first Flag is received. The MPSC never again enters the Hunt phase unless the microprocessor writes the Enter Hunt command. The MPSC will also detect flags separated by a single zero. For example, the bit pattern 011111101111110 will be detected as two flags.

The MPSC can be programmed to receive all frames or it can be programmed to the Address Search Mode. In the Address Search Mode, only frames with addresses that match the value in WR6 or the global address (0FFH) are received by the MPSC. Extended address recognition must be done by the microprocessor software.

The control and information fields are received as data.

SDLC/HDLC CRC calculation does not have an 8-bit delay, since all characters are included in the calculation, unlike Byte Synchronous Protocols.

Reception of an abort sequence (7 or more 1's) will cause the Break/Abort bit (RR0; D7) to be set and will cause an External/Status interrupt, if enabled. After the Reset External/Status Interrupts Command has been issued, a second interrupt will occur at the end of the abort sequence.

MPSC

Detailed Command/Status Description

GENERAL

The MPSC supports an extremely flexible set of serial and system interface modes.

The system interface to the CPU consists of 8 ports or buffers:

$\overline{CS}$	A_1	A_0	Read Operation	Write Operation
0	0	0	Ch. A Data Read	Ch. A Data Write
0	1	0	Ch. A Status Read	Ch. A Command/Parameter
0	0	1	Ch. B Data Read	Ch. B Data Write
0	1	1	Ch. B Status Read	Ch. B Command/Parameter
1	X	X	High Impedance	High Impedance

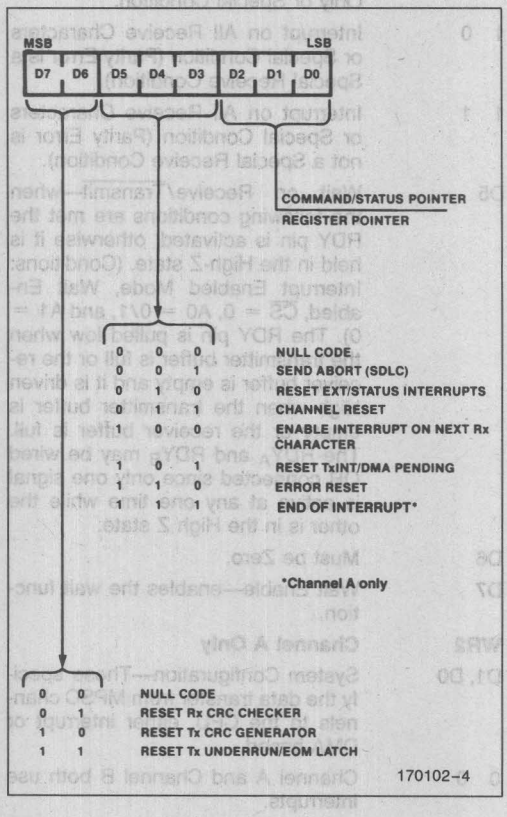
Data buffers are addressed by $A_1 = 0$, and Command ports are addressed by $A_1 = 1$.

COMMAND/STATUS DESCRIPTION

The following command and status bytes are used during initialization and execution phases of operation. All Command/Status operations on the two channels are identical, and independent, except where noted.

Detailed Register Description

Write Register 0 (WR0):



WR0

D2, D1, D0—Command/Status Register Pointer bits determine which write-register the next byte is to be written into, or which read-register the next byte is to be read from. After reset, the first byte written into either channel goes into WR0. Following a read or write to any register (except WR0) the pointer will point to WR0.

D5, D4, D3—Command bits determine which of the basic seven commands are to be performed.

Command 0 Null—has no effect.

Command 1 Send Abort—causes the generation of eight to thirteen 1's when in the SDLC mode.

Command 2 Reset External/Status Interrupts—resets the latched status bits of RR0 and re-enables them, allowing interrupts to occur again.

Command 3 Channel Reset—resets the Latched Status bits of RR0, the interrupt prioritization logic and all control registers for the channel. Four extra system clock cycles should be allowed for MPSC reset time before any additional commands or controls are written into the channel.

Command 4 Enable Interrupt on Next Receive Character—if the Interrupt on First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the MPSC for the next message.

Command 5 Reset Transmitter Interrupt/DMA Pending—if The Transmit Interrupt/DMA Enable mode is selected, the MPSC automatically interrupts or requests DMA data transfer when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts or DMA requests until the next character has been completely sent.

Command 6 Error Reset—error latches, Parity and Overrun errors in RR1 are reset.

Command 7 End of Interrupt—resets the interrupt-in-service latch of the highest-priority internal device under service.

D7, D6 CRC Reset Code.

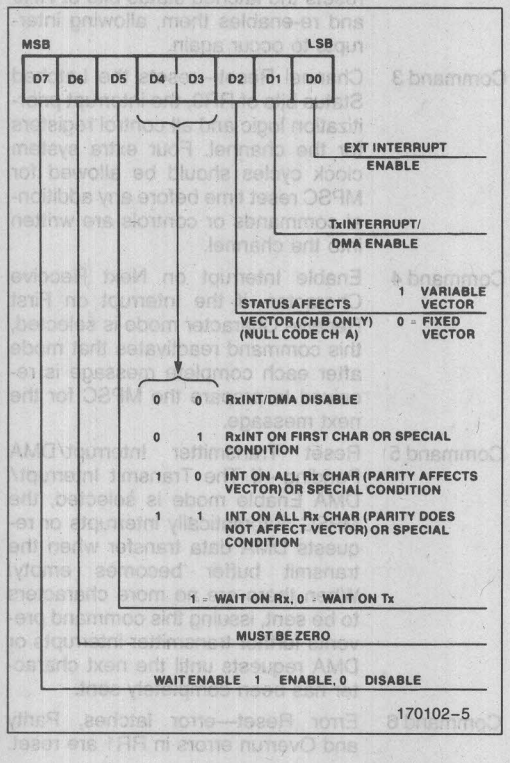
00 Null—has no effect.

01 Reset Receive CRC Checker—resets the CRC checker to 0's. If in SDLC mode the CRC checker is initialized to all 1's.

10 Reset Transmit CRC Generator—resets the CRC generator to 0's. If in SDLC mode the CRC generator's initialized to all 1's.

11 Reset Tx Underrun/End of Message Latch.

Write Register 1 (WR1):



WR1

D0 External/Status Interrupt Enable—allows interrupt to occur as the result of transitions on the $\overline{CD}$, $\overline{CTS}$ or $\overline{SYNDET}$ inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.

D1 Transmitter Interrupt/DMA Enable—allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

D2 Status Affects vector—(WR1, D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit if set then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

D4, D3 Receive Interrupt Mode.

0 0 Receive Interrupts/DMA Disabled.

0 1 Receive Interrupt on First Character Only or Special Condition.

1 0 Interrupt on All Receive Characters or Special Condition (Parity Error is a Special Receive Condition).

1 1 Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).

D5 Wait on Receive/$\overline{Transmit}$—when the following conditions are met the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, $\overline{CS}$ = 0, A0 = 0/1, and A1 = 0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDY_A and RDY_B may be wired OR connected since only one signal is active at any one time while the other is in the High Z state.

D6 Must be Zero.

D7 Wait Enable—enables the wait function.

WR2 **Channel A Only**

D1, D0 System Configuration—These specify the data transfer from MPSC channels to the CPU, either interrupt or DMA based.

0 0 Channel A and Channel B both use interrupts.

0 1 Channel A uses DMA, Channel B uses interrupts.

1 0 Channel A and Channel B both use DMA.

1 1 Illegal Code.

D2 Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.

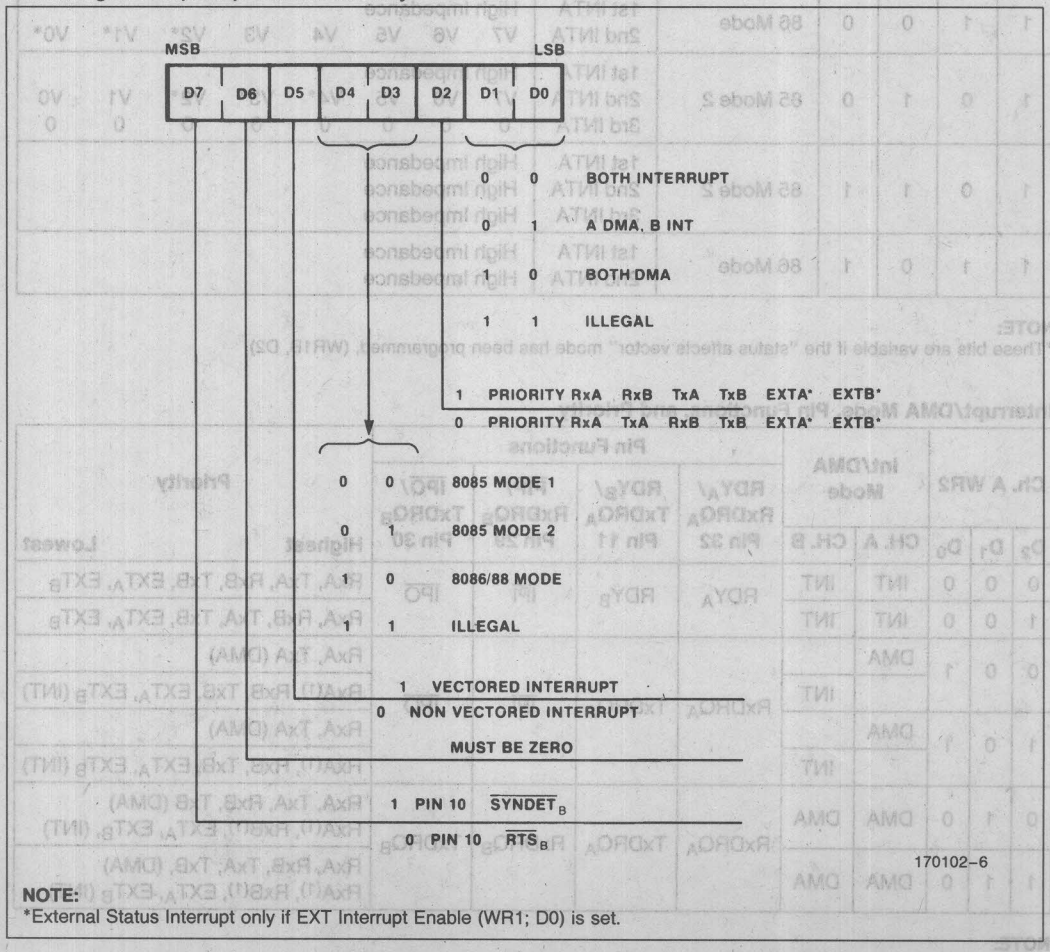
0 (Highest) RxA, TxA, RxB, TxB, ExTA, ExTB (Lowest).

1 (Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest).

D5, D4, D3 Interrupt Code—specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table.)

0 X X	Non-vectored interrupts—intended for use with external DMA CONTROLLER. The Data Bus remains in a high impedance state during INTA sequences.
1 0 0	8085 Vector Mode 1—intended for use as the primary MPSC in a daisy chained priority structure. (See System Interface section).
1 0 1	8085 Vector Mode 2—intended for use as any secondary MPSC in a daisy chained priority structure. (See System Interface section).
1 1 0	8086/88 Vector Mode—intended for use as either a primary or secondary in a daisy chained priority structure. (See System Interface section).
D6	Must be zero.
D7	zero Pin 10 = $\overline{RTS}_B$ one Pin 10 = $\overline{SYNDET}_B$

Write Register 2 (WR2): Channel A Only



NOTE:
*External Status Interrupt only if EXT Interrupt Enable (WR1; D0) is set.

The following table describes the MPSC's response to an interrupt acknowledge sequence:

D5	D4	D3	$\overline{IPI}$	MODE	INTA	Data Bus
0	X	X	X	Non-vectored	Any INTA	D7 ... D0 High Impedance
1	0	0	0	85 Mode 1	1st INTA 2nd INTA 3rd INTA	1 1 0 0 1 1 0 1 V7 V6 V5 V4* V3* V2* V1 V0 0 0 0 0 0 0 0 0
1	0	0	1	85 Mode 1	1st INTA 2nd INTA 3rd INTA	1 1 0 0 1 1 0 1 High Impedance High Impedance
1	1	0	0	86 Mode	1st INTA 2nd INTA	High Impedance V7 V6 V5 V4 V3 V2* V1* V0*
1	0	1	0	85 Mode 2	1st INTA 2nd INTA 3rd INTA	High Impedance V7 V6 V5 V4* V3* V2* V1 V0 0 0 0 0 0 0 0 0
1	0	1	1	85 Mode 2	1st INTA 2nd INTA 3rd INTA	High Impedance High Impedance High Impedance
1	1	0	1	86 Mode	1st INTA 2nd INTA	High Impedance High Impedance

NOTE:
*These bits are variable if the "status affects vector" mode has been programmed, (WR1B, D2).

Interrupt/DMA Mode, Pin Functions, and Priority

Ch. A WR2			Int/DMA Mode		Pin Functions				Priority
					RDY_A/ $RxDRQ_A$ Pin 32	RDY_B/ $TxDRQ_A$ Pin 11	$\overline{PIP}$/ $RxDRQ_B$ Pin 29	$\overline{IPO}$/ $TxDRQ_B$ Pin 30	Highest ... Lowest
D_2	D_1	D_0	CH. A	CH. B					
0	0	0	INT	INT	RDY_A	RDY_B	$\overline{IPI}$	$\overline{IPO}$	RxA, TxA, RxB, TxB, EXT_A, EXT_B
1	0	0	INT	INT					RxA, RxB, TxA, TxB, EXT_A, EXT_B
0	0	1	DMA		$RxDRQ_A$	$TxDRQ_A$	$\overline{IPI}$	$\overline{IPO}$	RxA, TxA (DMA)
				INT					RxA(1), RxB, TxB, EXT_A, EXT_B (INT)
1	0	1	DMA						RxA, TxA (DMA)
				INT					RxA(1), RxB, TxB, EXT_A, EXT_B (INT)
0	1	0	DMA	DMA	$RxDRQ_A$	$TxDRQ_A$	$RxDRQ_B$	$TxDRQ_B$	RxA, TxA, RxB, TxB (DMA) RxA(1), RxB(1), EXT_A, EXT_B, (INT)
1	1	0	DMA	DMA					RxA, RxB, TxA, TxB, (DMA) RxA(1), RxB(1), EXT_A, EXT_B (INT)

NOTE:
1. Special Receive Condition

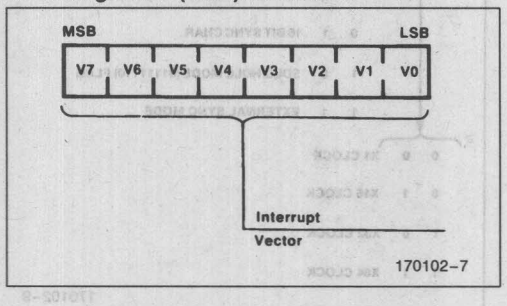
Interrupt Vector Mode Table

8085 Modes 8086/88 Mode	V_4 V_2	V_3 V_1	V_2 V_0	Channel	Condition
	0	0	0	B	Tx Buffer Empty
	0	0	1		Ext/Status Change
	0	1	0		Rx Char. Available
	0	1	1		Special Rx Condition (Note 1)
	1	0	0	A	Tx Buffer Empty
	1	0	1		Ext/Status Change
	1	1	0		Rx Char. Available
	1	1	1		Special Rx Condition (Note 1)

NOTE:
1. Special Receive Condition = Parity Error, Rx Overrun Error, Framing Error, End of Frame (SDLC).

2

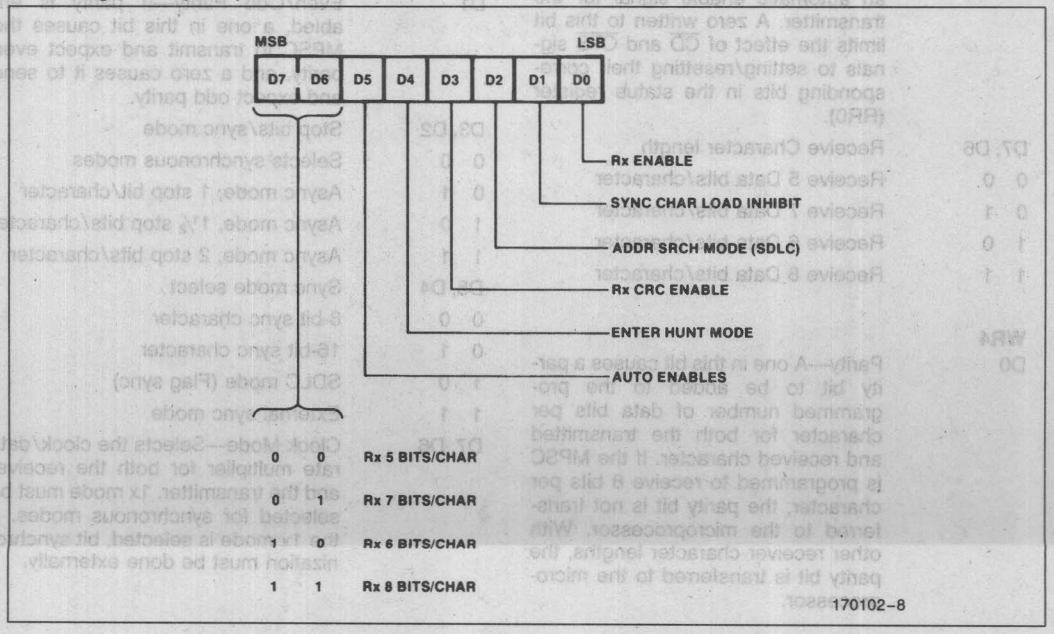
Write Register 2 (WR2): Channel B



WR2 CHANNEL B

D7–D0 Interrupt vector—This register contains the value of the interrupt vector placed on the data bus during interrupt acknowledge sequences.

Write Register 3 (WR3):



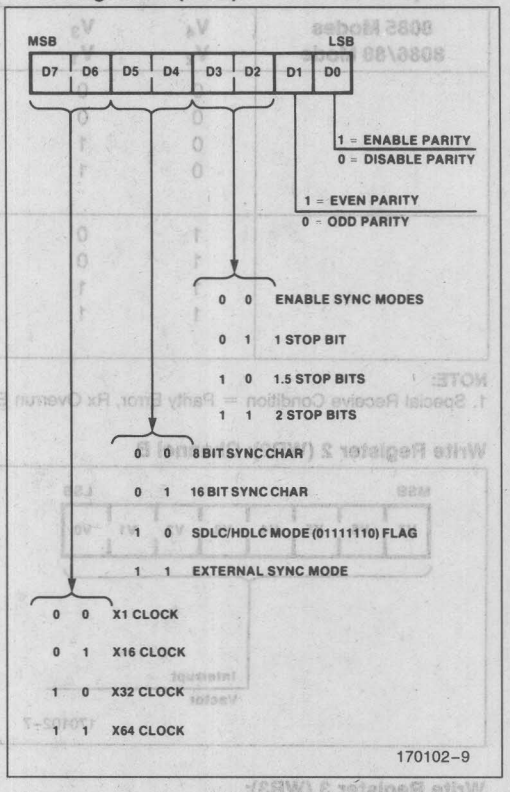
WR3

D0	Receiver Enable—A one enables the receiver to begin. This bit should be set only after the receiver has been initialized.
D1	Sync Character Load Inhibit—A one prevents the receiver from loading sync characters into the receive buffers. In SDLC, this bit must be zero.
D2	Address Search Mode—If the SDLC mode has been selected, the MPSC will receive all frames unless this bit is a 1. If this bit is a 1, the MPSC will receive only frames with address (0FFH) or the value loaded into WR6. This bit must be zero in non-SDLC modes.
D3	Receive CRC Enable—A one in this bit enables (or re-enables) CRC calculation. CRC calculation starts with the last character placed in the Receiver FIFO. A zero in this bit disables, but does not reset, the Receiver CRC generator.
D4	Enter Hunt Phase—After initialization, the MPSC automatically enters the Hunt mode. If synchronization is lost, the Hunt phase can be re-entered by writing a one to this bit.
D5	Auto Enable—A one written to this bit causes $\overline{CD}$ to be automatic enable signal for the receiver and $\overline{CTS}$ to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of $\overline{CD}$ and $\overline{CTS}$ signals to setting/resetting their corresponding bits in the status register (RR0).
D7, D6	Receive Character length
0 0	Receive 5 Data bits/character
0 1	Receive 7 Data bits/character
1 0	Receive 6 Data bits/character
1 1	Receive 8 Data bits/character

WR4

D0	Parity—A one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.

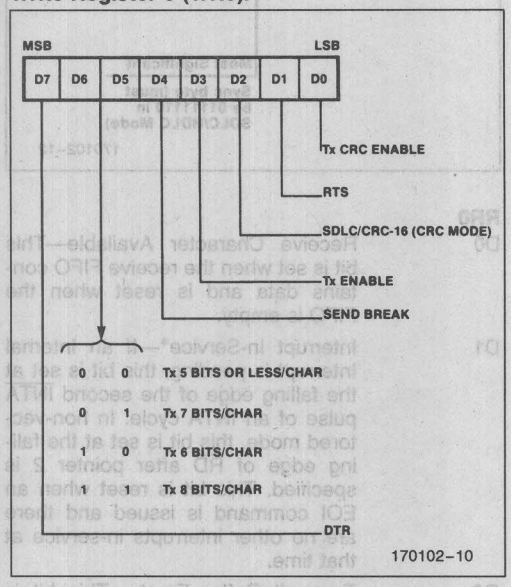
Write Register 4 (WR4):



D1	Even/Odd Parity—If parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and a zero causes it to send and expect odd parity.
D3, D2	Stop bits/sync mode
0 0	Selects synchronous modes
0 1	Async mode, 1 stop bit/character
1 0	Async mode, 1½ stop bits/character
1 1	Async mode, 2 stop bits/character
D5, D4	Sync mode select
0 0	8-bit sync character
0 1	16-bit sync character
1 0	SDLC mode (Flag sync)
1 1	External sync mode
D7, D6	Clock Mode—Selects the clock/data rate multiplier for both the receiver and the transmitter. 1x mode must be selected for synchronous modes. If the 1x mode is selected, bit synchronization must be done externally.

0	0	Clock rate = Data rate × 1
0	1	Clock rate = Data rate × 16
1	0	Clock rate = Data rate × 32
1	1	Clock rate = Data rate × 64

Write Register 5 (WR5):



WR5

D0 Transmit CRC Enable—A one in this bit enables the transmitter CRC generator. The CRC calculation is done when a character is moved from the transmit buffer into the shift register. A zero in this bit disables CRC calculations. If this bit is not set when a transmitter underrun occurs, the CRC will not be sent.

D1 Request to Send—A one in this bit forces the $\overline{RTS}$ pin active (low) and zero in this bit forces the $\overline{RTS}$ pin inactive (high).

D2 CRC Select—A one in this bit selects the CRC−16 polynomial ($X^{16} + X^{15} + X^2 + 1$) and a zero in this bit selects the CCITT-CRC polynomial ($X^{16} + X^{12} + X^5 + 1$). In SDLC mode, CCITT-CRC must be selected.

D3 Transmitter Enable—A zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.

D4 Send Break—A one in this bit forces the transmit data low. A zero in this bit allows normal transmitter operation.

D6, D5 Transmit Character length

0	0	Transmit 1–5 bits/character
0	1	Transmit 7 bits/character
1	0	Transmit 6 bits/character
1	1	Transmit 8 bits/character

Bits to be sent must be right justified least significant bit first, e.g.:

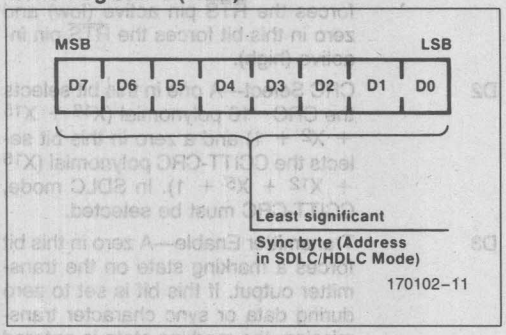
D7	D6	D5	D4	D3	D2	D1	D0
0	0	B5	B4	B3	B2	B1	B0

D7 Data Terminal Ready—When set, this bit forces the $\overline{DTR}$ pin active (low). When reset, this bit forces the $\overline{DTR}$ pin inactive (high).

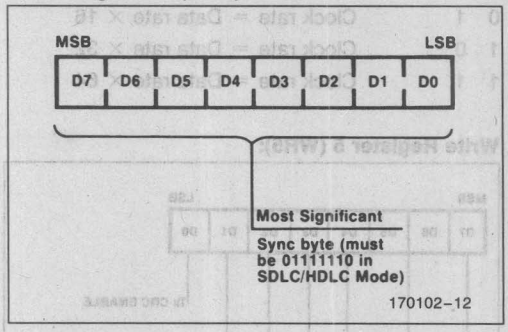
Five or less mode allows transmission of one to five bits per character. The microprocessor must format the data in the following way:

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	1	0	0	0	B0	Sends one data bit
1	1	1	0	0	0	B1	B0	Sends two data bits
1	1	0	0	0	B2	B1	B0	Sends three data bits
1	0	0	0	B3	B2	B1	B0	Sends four data bits
0	0	0	B4	B3	B2	B1	B0	Sends five data bits

Write Register 6 (WR6):



Write Register 7 (WR7):



WR6

D7–D0 Sync/Address—This register contains the transmit sync character in Monosync mode, the low order 8 sync bits in Bisync mode, or the Address byte in SDLC mode.

WR7

D7–D0 Sync/Flag—This register contains the receive sync character in Monosync mode, the high order 8 sync bits in Bisync mode, or the Flag character (01111110) in SDLC mode. WR7 is not used in External Sync mode.

RR0

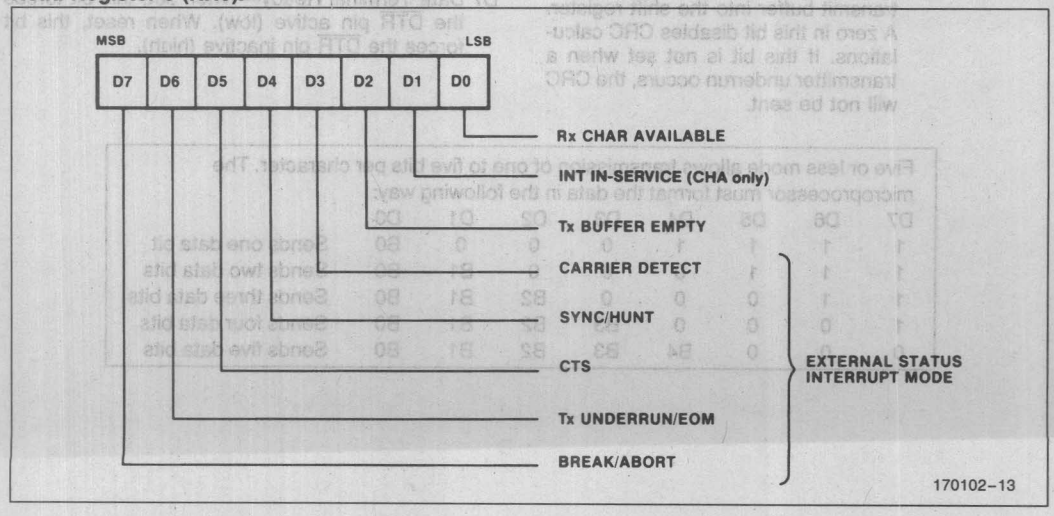
D0 Receive Character Available—This bit is set when the receive FIFO contains data and is reset when the FIFO is empty.

D1 Interrupt In-Service*—If an Internal Interrupt is pending, this bit is set at the falling edge of the second $\overline{INTA}$ pulse of an INTA cycle. In non-vectored mode, this bit is set at the falling edge of $\overline{RD}$ after pointer 2 is specified. This bit is reset when an EOI command is issued and there are no other interrupts in-service at that time.

D2 Transmit Buffer Empty—This bit is set whenever the transmit buffer is

*This bit is only valid when $\overline{IPI}$ is active low and is always zero in Channel B.

Read Register 0 (RR0):



empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.

D3 Carrier Detect—This bit contains the state of the $\overline{CD}$ pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the $\overline{CD}$ pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the $\overline{CD}$ pin immediately following a Reset External/Status Interrupt command.

D4 Sync/Hunt—In asynchronous modes, the operation of this bit is similar to the CD status bit, except that Sync/Hunt shows the state of the $\overline{SYNDET}$ input. Any High-to-Low transition on the $\overline{SYNDET}$ pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the $\overline{SYNDET}$ pin at time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the $\overline{SYNDET}$ input.

In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the $\overline{SYNDET}$ input must be held High by the external logic until external character synchronization is achieved. A High at the $\overline{SYNDET}$ input holds the Sync/Hunt status in the reset condition.

When external synchronization is achieved, $\overline{SYNDET}$ must be driven Low on the second rising edge of $\overline{RxC}$ after the rising edge of $\overline{RxC}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activitate the $\overline{SYNDET}$ input. Once $\overline{SYNDET}$ is forced Low, it is good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. The High-to-Low transition of the $\overline{SYNDET}$ output sets the Sync/Hunt bit, which sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt Command.

When the $\overline{SYNDET}$ input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the MPSC again looks for a High-to-Low transition on the $\overline{SYNDET}$ input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the MPSC is waiting for $\overline{SYNDET}$ to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the MPSC establishes character synchronization. The High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the MPSC to detect the next transition of other External/Status bits.

When the CPU detects the end of message or that character synchronization is lost, it sets the Enter Hunt Mode control bit, which sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status Interrupt, which must also be cleared by the Reset External/Status Interrupt Command. Note that the $\overline{SYNDET}$ pin acts as an output in this mode, and goes low every time a sync pattern is detected in the data stream.

In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit, or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the MPSC. The External/Status interrupt is also generated, and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The MPSC automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the receiver.

D5 — Clear to Send—This bit contains the inverted state of the $\overline{CTS}$ pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the $\overline{CTS}$ pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the $\overline{CTS}$ pin immediately following a Reset External/Status Interrupt command.

D6 — Transmitter Underrun/End of Message—This bit is in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D_6 and D_7). When the Transmit Underrun condition occurs, this bit is set, which causes the External/Status Interrupt which must be reset by issuing a Reset External/Status command (WR0; command 2).

D7 — Break/Abort—In the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, Command 2) to the break detection logic so the Break sequence termination can be recognized.

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

D0 — All Sent—This bit is set when all characters have been sent, in asynchronous modes. It is reset when characters are in the transmitter, in asynchronous modes. In synchronous modes, this bit is always set.

RR1:

D3, D2, D1 — Residue Codes—Bit synchronous protocols allow I-fields that are not an integral number of characters. Since transfers from the MPSC to the CPU are character oriented, the residue codes provide the capability of receiving leftover bits. Residue bits are right justified in the last data byte received or first CRC byte.

D4 — Parity Error—If parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

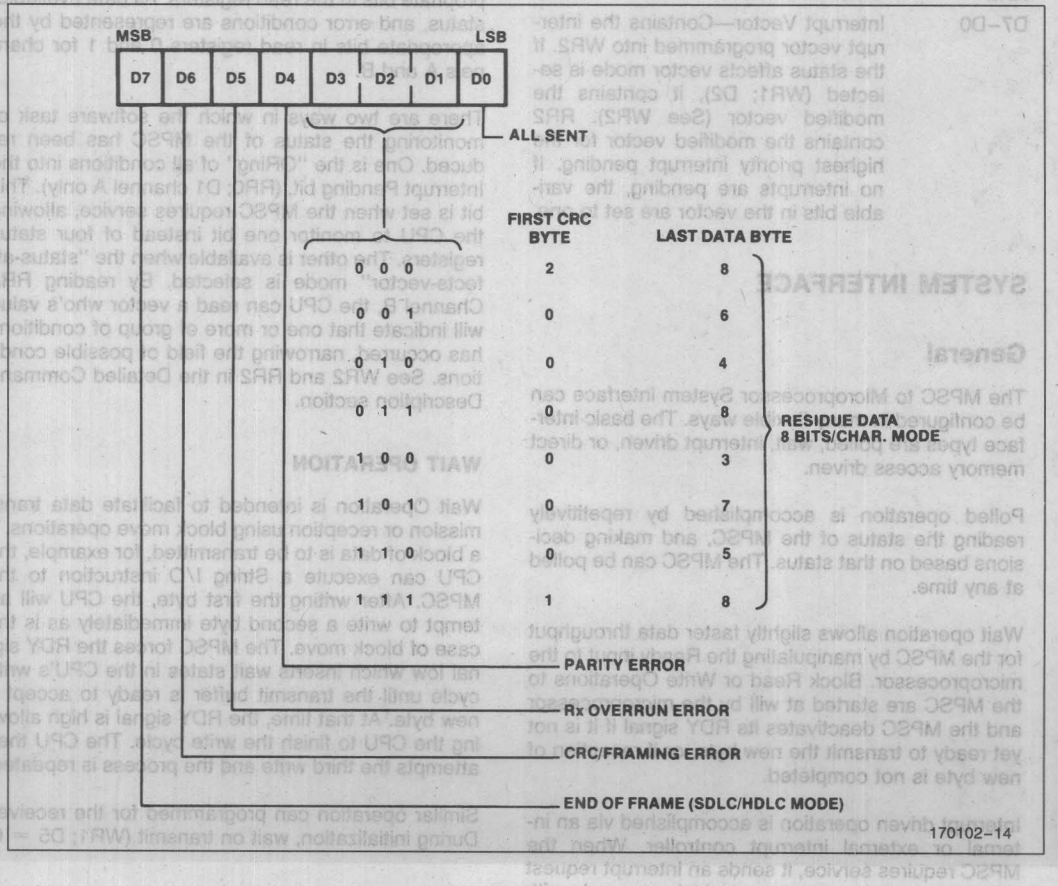
D5 — Receive Overrun Error—This bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwritten and flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the status affects vector mode, the overrun causes a special Receive Condition Vector.

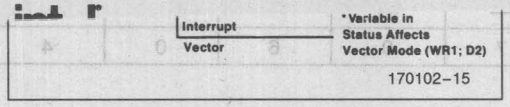
D6 — CRC/Framing Error—In async modes, a one in this bit indicates a receive framing error. In synchronous modes, a one in this bit indicates that the calculated CRC value does not match the last two bytes received. It can be reset by issuing an Error Reset command.

SDLC Residue Code Table (I Field Bits in 2 Previous Bytes)

RR1			8 bits/char		7 bits/char		6 bits/char		5 bits/char	
D3	D2	D1	First CRC Byte	Last Data Byte	First CRC Byte	Last Data Byte	First CRC Byte	Last Data Byte	First CRC Byte	Last Data Byte
1	0	0	0	3	0	2	0	1	0	5
0	1	0	0	4	0	3	0	2	0	1
1	1	0	0	5	0	4	0	3	0	2
0	0	1	0	6	0	5	0	4	0	3
1	0	1	0	7	0	6	0	5	—	—
0	1	1	0	8	0	—	—	—	—	—
1	1	1	1	8	—	—	—	—	—	—
0	0	0	2	8	1	7	0	6	0	4

Read Register 1 (RR1): (Special Receive Condition Mode)





RR2	Channel B
D7–D0	Interrupt Vector—Contains the interrupt vector programmed into WR2. If the status affects vector mode is selected (WR1; D2), it contains the modified vector (See WR2). RR2 contains the modified vector for the highest priority interrupt pending. If no interrupts are pending, the variable bits in the vector are set to one.

SYSTEM INTERFACE

General

The MPSC to Microprocessor System interface can be configured in many flexible ways. The basic interface types are polled, wait, interrupt driven, or direct memory access driven.

Polled operation is accomplished by repetitively reading the status of the MPSC, and making decisions based on that status. The MPSC can be polled at any time.

Wait operation allows slightly faster data throughput for the MPSC by manipulating the Ready input to the microprocessor. Block Read or Write Operations to the MPSC are started at will by the microprocessor and the MPSC deactivates its RDY signal if it is not yet ready to transmit the new byte, or if reception of new byte is not completed.

Interrupt driven operation is accomplished via an internal or external interrupt controller. When the MPSC requires service, it sends an interrupt request signal to the microprocessor, which responds with

POLLED OPERATION

In the polled mode, the CPU must monitor the desired conditions within the MPSC by reading the appropriate bits in the read registers. All data available, status, and error conditions are represented by the appropriate bits in read registers 0 and 1 for channels A and B.

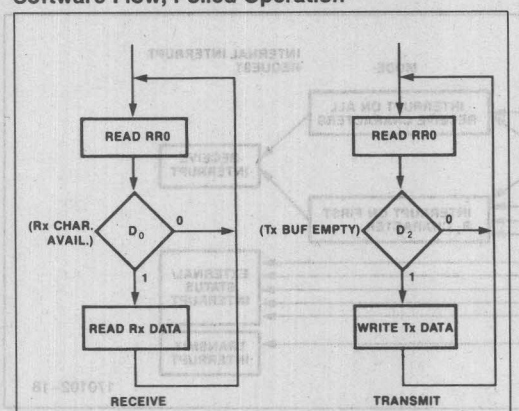
There are two ways in which the software task of monitoring the status of the MPSC has been reduced. One is the "ORing" of all conditions into the Interrupt Pending bit. (RR0; D1 channel A only). This bit is set when the MPSC requires service, allowing the CPU to monitor one bit instead of four status registers. The other is available when the "status-affects-vector" mode is selected. By reading RR2 Channel B, the CPU can read a vector who's value will indicate that one or more of group of conditions has occurred, narrowing the field of possible conditions. See WR2 and RR2 in the Detailed Command Description section.

WAIT OPERATION

Wait Operation is intended to facilitate data transmission or reception using block move operations. If a block of data is to be transmitted, for example, the CPU can execute a String I/O instruction to the MPSC. After writing the first byte, the CPU will attempt to write a second byte immediately as is the case of block move. The MPSC forces the RDY signal low which inserts wait states in the CPU's write cycle until the transmit buffer is ready to accept a new byte. At that time, the RDY signal is high allowing the CPU to finish the write cycle. The CPU then attempts the third write and the process is repeated.

Similar operation can programmed for the receiver. During initialization, wait on transmit (WR1; D5 = 0)

Software Flow, Polled Operation



170102-16

NOTES:
1. RR0; D0 is reset automatically when the data is read.
2. RR0; D2 is reset automatically when the data is written.

or wait on receive (WR1; D5 = 1) can be selected. The wait operation can be enabled/disabled by setting/resetting the Wait Enable Bit (WR1; D7).

NOTE:
CAUTION: ANY CONDITION THAT CAN CAUSE THE TRANSMITTER TO STOP (E.G., CTS GOES INACTIVE) OR THE RECEIVER TO STOP (E.G., RX DATA STOPS) WILL CAUSE THE MPSC TO HANG THE CPU UP IN WAIT STATES UNTIL RESET. EXTREME CARE SHOULD BE TAKEN WHEN USING THIS FEATURE.

INTERRUPT DRIVEN OPERATION

The MPSC can be programmed into several interrupt modes: Non-Vectored, 8085 vectored, and 8088/86 vectored. In both vectored modes, multiple MPSC's can be daisy-chained.

In the vectored mode, the MPSC responds to an interrupt acknowledge sequence by placing a call instruction (8085 mode) and interrupt vector (8085 and 8088/86 mode) on the data bus.

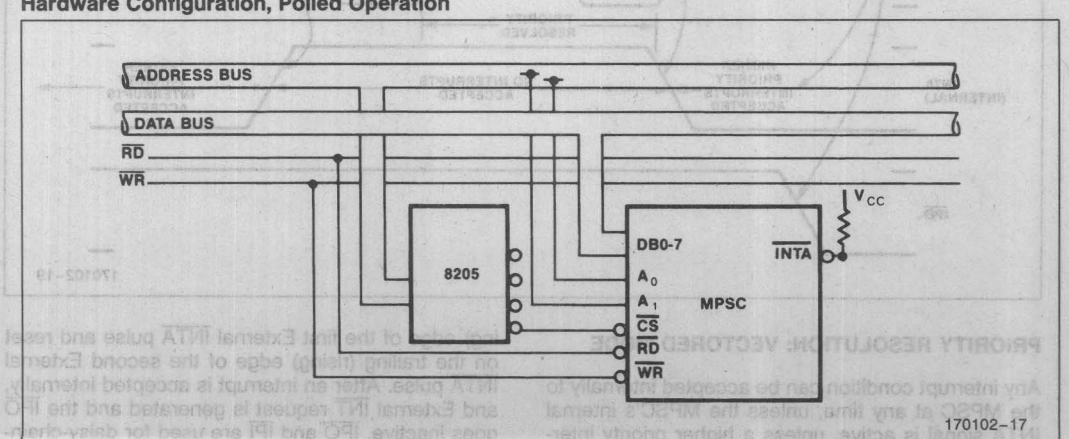
The MPSC can be programmed to cause an interrupt due to up to 14 conditions in each channel. The status of these interrupt conditions is contained in Read Registers 0 and 1. These 14 conditions are all directed to cause 3 different types of internal interrupt request for each channel: receive/interrupts, transmit interrupts and external/status interrupts (if enabled).

This results in up to 6 internal interrupt request signals. The priority of those signals can be programmed to one of two fixed modes:

Highest Priority					Lowest Priority
RxA	RxB	TxA	TxB	ExTA	ExTB
RxA	TxA	RxB	TxB	ExTA	ExTB

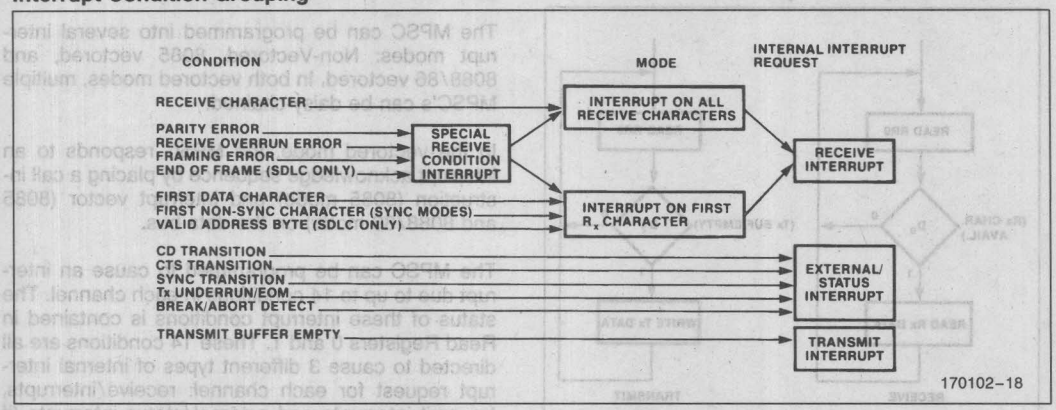
The interrupt priority resolution works differently for vectored and non-vectored modes.

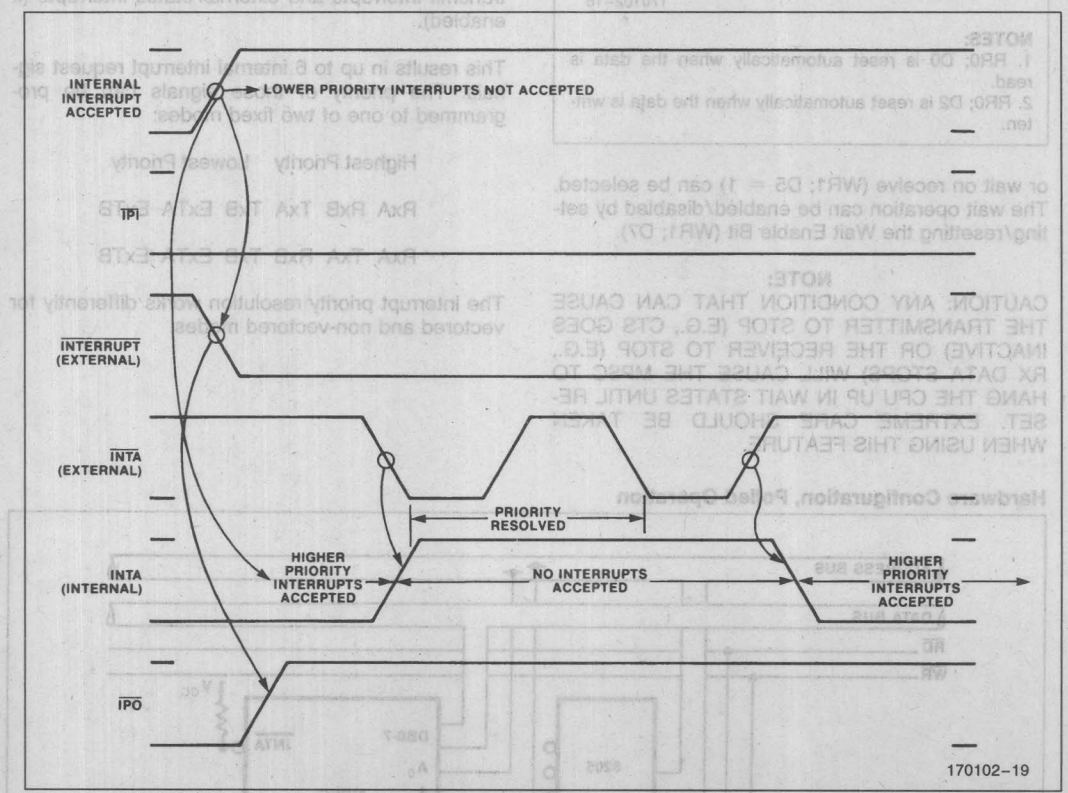
Hardware Configuration, Polled Operation



170102-17

Interrupt Condition Grouping

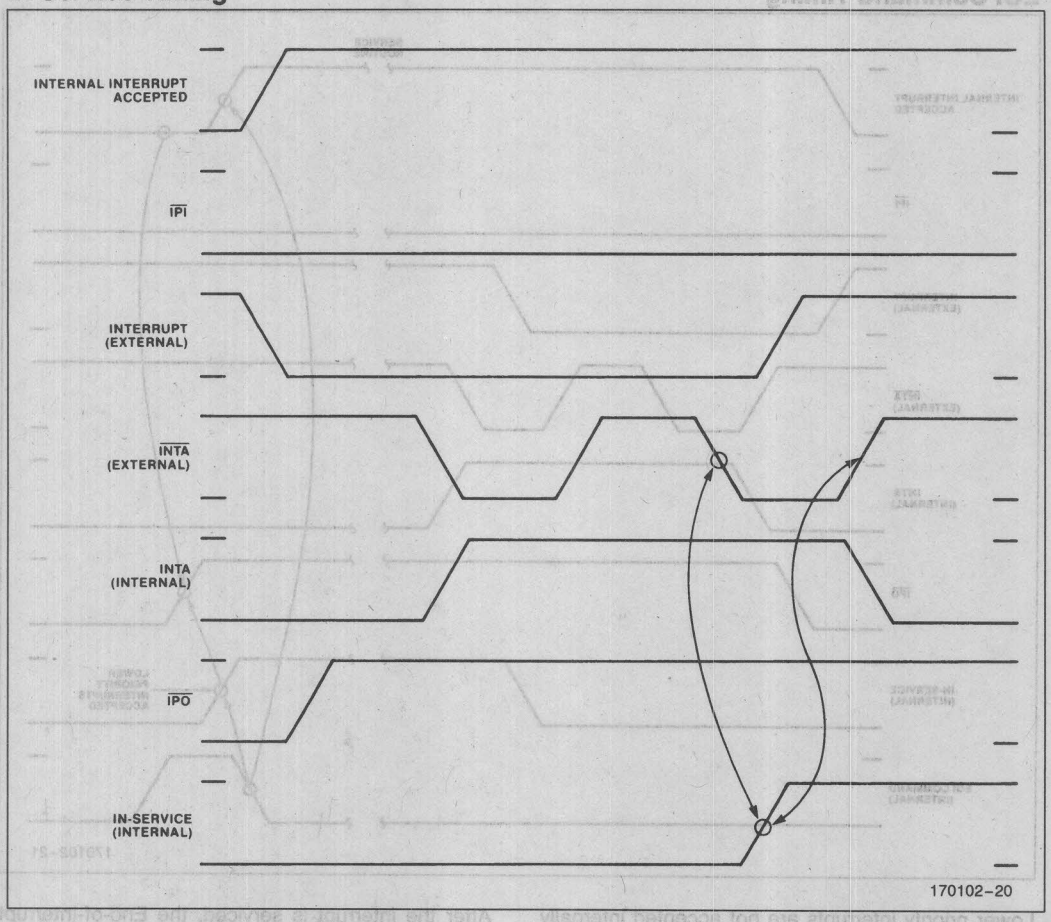




PRIORITY RESOLUTION: VECTORED MODE

Any interrupt condition can be accepted internally to the MPSC at any time, unless the MPSC's internal INTA signal is active, unless a higher priority interrupt is currently accepted, or if $\overline{IPI}$ is inactive (high). The MPSC's internal INTA is set on the leading (falling) edge of the first External $\overline{INTA}$ pulse and reset on the trailing (rising) edge of the second External $\overline{INTA}$ pulse. After an interrupt is accepted internally, and External $\overline{INT}$ request is generated and the $\overline{IPO}$ goes inactive. $\overline{IPO}$ and $\overline{IPI}$ are used for daisy-chaining MPSC's together.

In-Service Timing

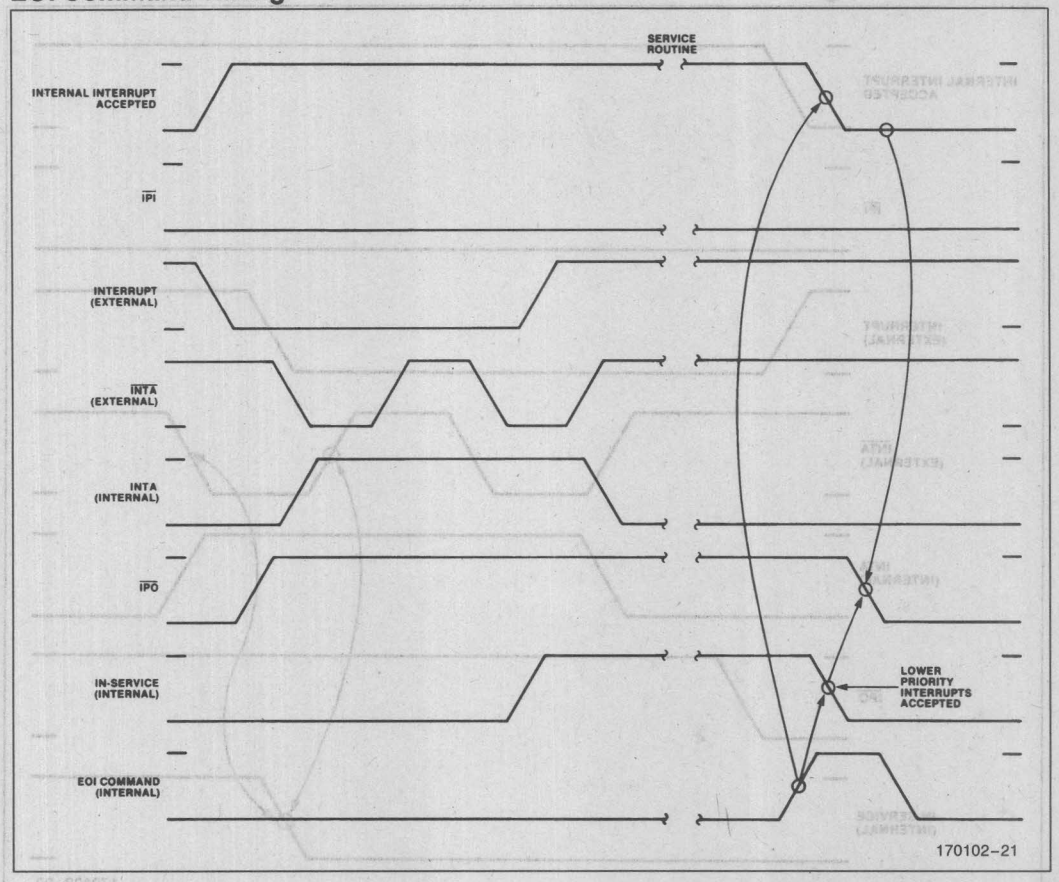


The MPSC's internal INTA is set on the leading (falling) edge of the first external $\overline{INTA}$ pulse, and reset on the trailing (rising) edge of the second external $\overline{INTA}$ pulse. After an interrupt is accepted internally, and external $\overline{INT}$ request is generated and $\overline{IPO}$ goes inactive (high). $\overline{IPO}$ and $\overline{IPI}$ are used for daisy-chaining MPSC's together.

Each of the six interrupt sources has an associated In-Service latch. After priority has been resolved, the highest priority In-Service latch is set. After the In-Service latch is set, the $\overline{INT}$ pin goes inactive (high).

NOTE:
If the External $\overline{INT}$ pin is active and the IPI signal is pulled inactive high, the $\overline{INT}$ signal will also go inactive. $\overline{IPI}$ qualifies the External $\overline{INT}$ Signal.

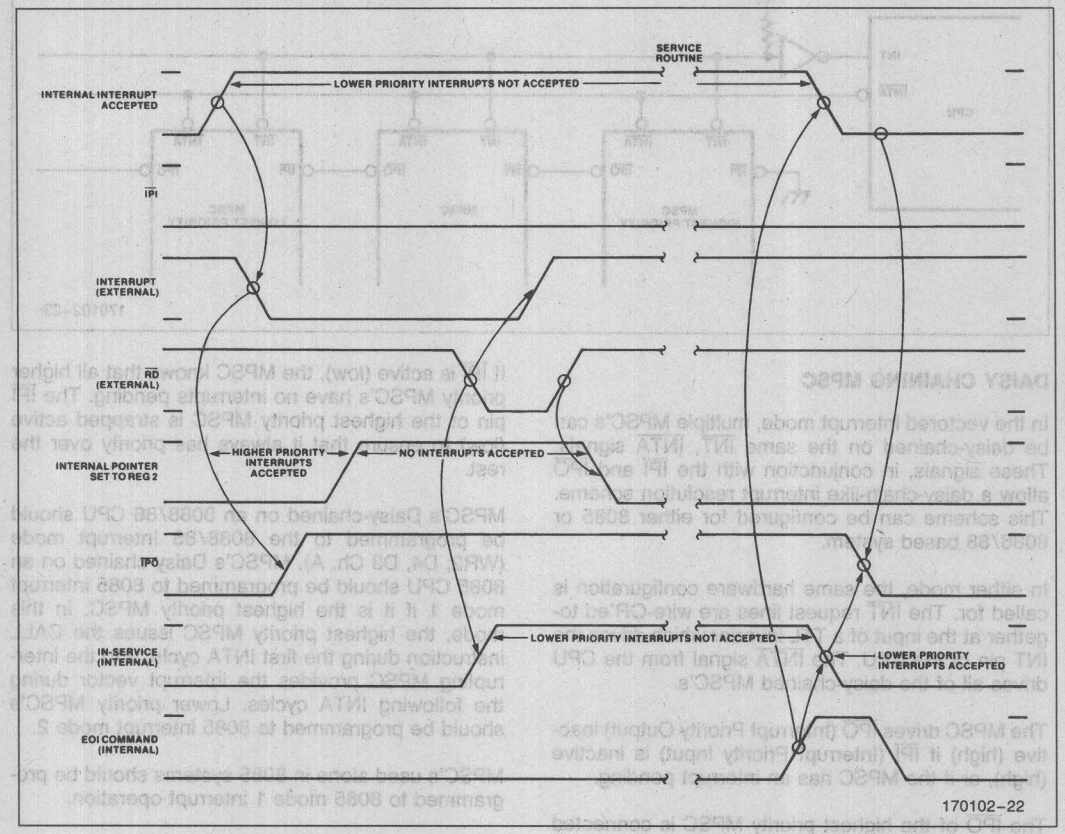
EOI Command Timing



Lower priority interrupts are not accepted internally while the In-Service latch is set. However, higher priority interrupts are accepted internally and a new external $\overline{INT}$ request is generated. If the CPU responds with a new INTA sequence, the MPSC will respond as before, suspending the lower priority interrupt.

After the interrupt is serviced, the End-of-Interrupt (EOI) command should be written to the MPSC. This command will cause an internal pulse that is used to reset the In-Service Latch which allows service for lower priority interrupts in the daisy-chain to resume, provided a new INTA sequence does not start for a higher priority interrupt (higher than the highest under service). If there is no interrupt pending internally, the $\overline{IPO}$ follows $\overline{IPI}$.

Non-Vectored Interrupt Timing

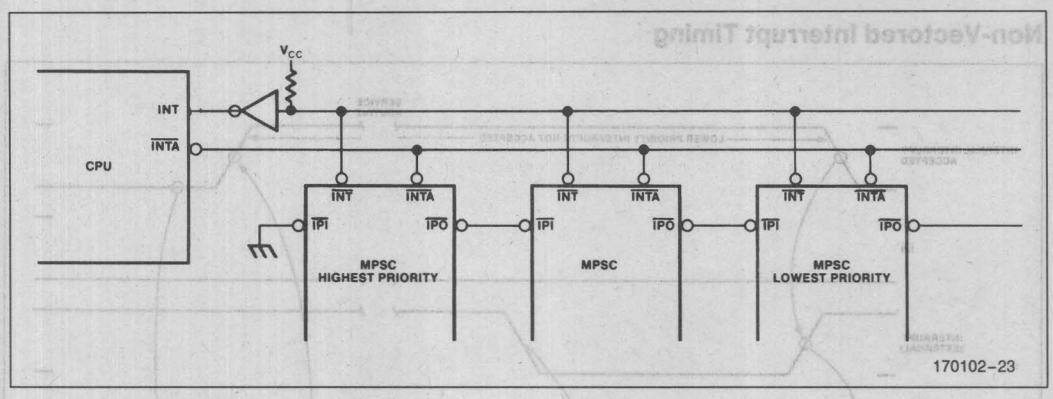


PRIORITY RESOLUTION: NON-VECTORED MODE

In non-vectored mode, the MPSC does not respond to interrupt acknowledge sequences. The $\overline{INTA}$ input (pin 27) must be pulled high for proper operation. The MPSC should be programmed to the Status-Affects-Vector mode, and the CPU should read RR2 (Ch. B) in its service routine to determine which interrupt requires service.

In this case, the internal pointer being set to RR2 provides the same function as the internal INTA signal in the vectored mode. It inhibits acceptance of any additional internal interrupts and its leading edge starts the interrupt priority resolution circuit. The interrupt priority resolution is ended by the leading edge of the read signal used by the CPU to retrieve the modified vector. The leading edge of read sets the In-Service latch and forces the external $\overline{INT}$ output inactive (high). The internal pointer is reset to zero after the trailing edge of the read pulse.

NOTE:
That if RR2 is specified but not read, no internal interrupts, regardless of priority, are accepted.



DAISY CHAINING MPSC

In the vectored interrupt mode, multiple MPSC's can be daisy-chained on the same $\overline{INT}$, $\overline{INTA}$ signals. These signals, in conjunction with the $\overline{IPI}$ and $\overline{IPO}$ allow a daisy-chain-like interrupt resolution scheme. This scheme can be configured for either 8085 or 8086/88 based system.

In either mode, the same hardware configuration is called for. The $\overline{INT}$ request lines are wire-OR'ed together at the input of a TTL inverter which drives the INT pin of the CPU. The $\overline{INTA}$ signal from the CPU drives all of the daisy-chained MPSC's.

The MPSC drives $\overline{IPO}$ (Interrupt Priority Output) inactive (high) if $\overline{IPI}$ (Interrupt Priority Input) is inactive (high), or if the MPSC has an interrupt pending.

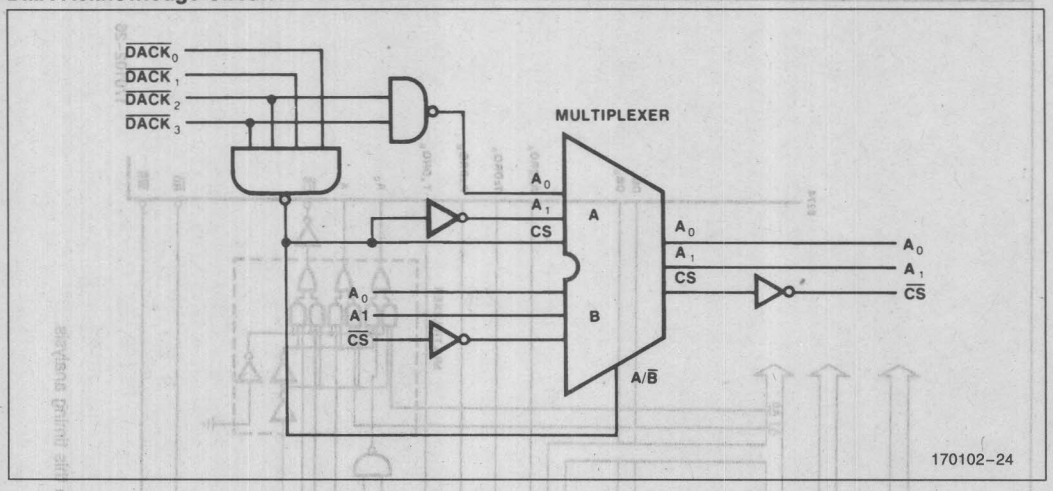
The $\overline{IPO}$ of the highest priority MPSC is connected to the $\overline{IPI}$ of the next highest priority MPSC, and so on.

If $\overline{IPI}$ is active (low), the MPSC knows that all higher priority MPSC's have no interrupts pending. The $\overline{IPI}$ pin of the highest priority MPSC is strapped active (low) to ensure that it always has priority over the rest.

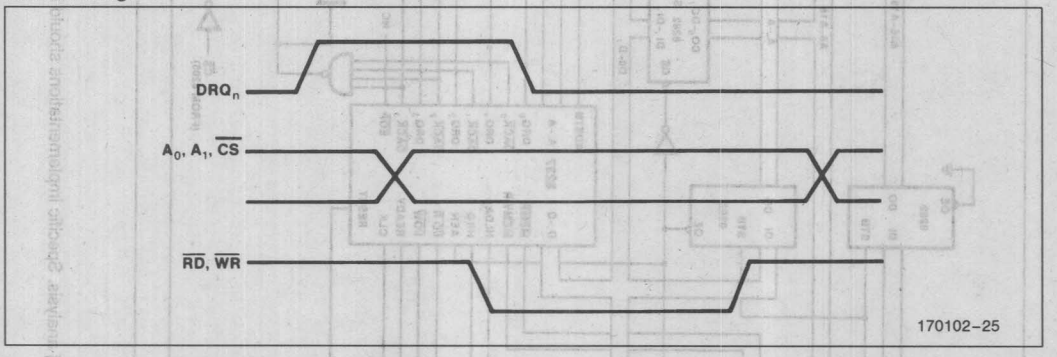
MPSC's Daisy-chained on an 8088/86 CPU should be programmed to the 8088/86 Interrupt mode (WR2; D4, D3 Ch. A). MPSC's Daisy-chained on an 8085 CPU should be programmed to 8085 interrupt mode 1 if it is the highest priority MPSC. In this mode, the highest priority MPSC issues the CALL instruction during the first INTA cycle, and the interrupting MPSC provides the interrupt vector during the following INTA cycles. Lower priority MPSC's should be programmed to 8085 interrupt mode 2.

MPSC's used alone in 8085 systems should be programmed to 8085 mode 1 interrupt operation.

DMA Acknowledge Circuit



DMA Timing



DMA OPERATION

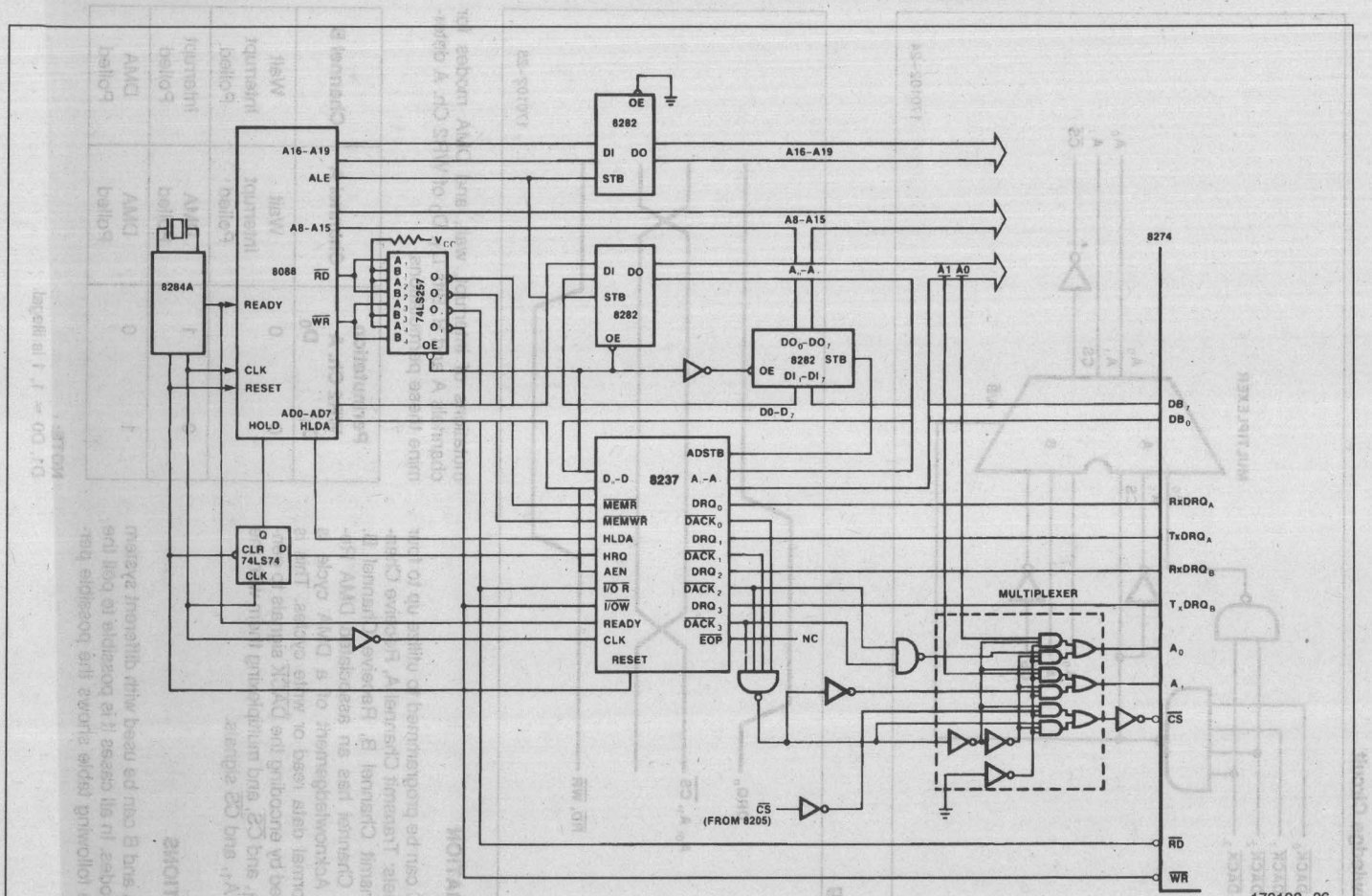
Each MPSC can be programmed to utilize up to four DMA channels: Transmit Channel A, Receive Channel A, Transmit Channel B, Receive Channel B. Each DMA Channel has an associated DMA Request line. Acknowledgement of a DMA cycle is done via normal data read or write cycles. This is accomplished by encoding the $\overline{DACK}$ signals to generate A_0, A_1, and $\overline{CS}$, and multiplexing them with the normal A_0, A_1, and $\overline{CS}$ signals.

PERMUTATIONS

Channels A and B can be used with different system interface modes. In all cases it is possible to poll the MPSC. The following table shows the possible permutations of interrupt, wait, and DMA modes for channels A and B. Bits D_1, D_0 of WR2 Ch. A determine these permutations.

Permutation WR2 Ch. A D_1	D_0	Channel A	Channel B
0	0	Wait Interrupt Polled	Wait Interrupt Polled
0	1	DMA Polled	Interrupt Polled
1	0	DMA Polled	DMA Polled

NOTE:
D1, D0 = 1, 1 is illegal.



170102-26

NOTE:
The circuit was not designed based on a worst-case timing analysis. Specific implementations should include this timing analysis.

PROGRAMMING HINTS

This section will describe some useful programming hints which may be useful in program development.

Asynchronous Operation

At the end of transmission, the CPU must issue "Reset Transmit Interrupt/DMA Pending" command in WR0 to reset the last transmit empty request which was not satisfied. Failing to do so will result in the MPSC locking up in a transmit empty state forever.

Non-Vectored Mode

In non-vectored mode, the Interrupt Acknowledge pin (INTA) on the MPSC must be tied high through a pull-up resistor. Failing to do so will result in unpredictable response from the 8274.

HDLC/SDLC Mode

When receiving data in SDLC mode, the CRC bytes must be read by the CPU (or DMA controller) just like any other data field. Failing to do so will result in receiver buffer overflow. The CRC bytes are not to be used for CRC verification. Residue bits may be contained in the first CRC byte. Also, the End of Frame Interrupt indicates that the entire frame has been received. At this point, the CRC result (RR1: D6) and residue code (RR1; D3, D2, D1) may be checked.

Status Register RR2

RR2 contains the vector which gets modified to indicate the source of interrupt (See the section titled MPSC Modes of Operation). However, the state of the vector does not change if no new interrupts are generated. The contents of RR2 are only changed when a new interrupt is generated. In order to get the correct information, RR2 must be read only after an interrrupt is generated, otherwise it will indicate the previous state.

Initialization Sequence

The MPSC initialization routine must issue a channel Reset Command at the beginning. WR4 should be defined before other registers. At the end of the initialization sequence, Reset External/Status and Error Reset commands should be issued to clear any spurious interrupts which may have been caused at power up.

Transmit Under-run/EOM Latch

In SDLC/HDLC, bisync and monosync mode, the transmit under-run/EOM must be reset to enable the CRC check bytes to be appended to the transmit frame or transmit message. The transmit under-run/EOM latch can be reset only after the first character is loaded into the transmit buffer. When the transmitter under-runs at the end of the frame, CRC check bytes are appended to the frame/message. The transmit under-run/EOM latch can be reset at any time during the transmission after the first character. However, it should be reset *before* the transmitter under-runs otherwise, both bytes of the CRC may not be appended to the frame/message. In the receive mode in bisync operation, the CPU must read the CRC bytes and two more SYNC characters before checking for valid CRC result in RR1.

2

Sync Character Load Inhibit

In bisync/monosync mode only, it is possible to prevent loading sync characters into the receive buffers by setting the sync character load inhibit bit (WR3; D1 = 1). Caution must be exercised in using this option. It may be possible to get a CRC character in the received message which may match the sync character and not get transferred to the receive buffer. However, sync character load inhibit should be enabled during all pre-frame sync characters so the software routine does not have to read them from the MPSC.

In SDLC/HDLC mode, sync character load inhibit bit must be reset to zero for proper operation.

EOI Command

EOI command can only be issued through channel A irrespective of which channel had generated the interrupt.

Priority in DMA Mode

There is no priority in DMA mode between the following four signals: TxDRQ(CHA), RxDRQ(CHA), TxDRQ(CHB), RxDRQ(CHB). The priority between these four signals must be resolved by the DMA controller. At any given time, all four DMA channels from the 8274 are capable of going active.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature
Under Bias . . . . . 0°C to +70°C
Storage Temperature
(Ceramic Package) . . . . . −65°C to +150°C
(Plastic Package) . . . . . −40°C to +125°C
Voltage on Any Pin with
Respect to Ground . . . . . −0.5V to +7.0V

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to +70°C; V_{CC} = +5V ±10%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC} +0.5	V	
V_{OL}	Output Low Voltage		+0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = 200 μA
I_{IL}	Input Leakage Current		±10	μA	V_{IN} = V_{CC} to 0V
I_{OFL}	Output Leakage Current		±10	μA	V_{OUT} = V_{CC} to 0.45V
I_{CC}	V_{CC} Supply Current		200	mA	

NOTE:
1. For Extended Temperature EXPRESS, use MIL8274 electrical Parameters.

CAPACITANCE T_A = 25°C; V_{CC} = GND = 0V

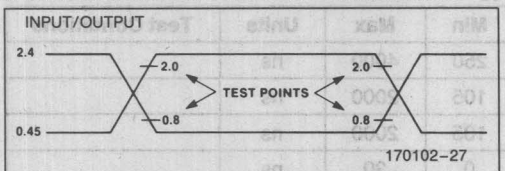
Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
C_{OUT}	Output Capacitance		15	pF	Unmeasured pins returned to GND
$C_{I/O}$	Input/Output Capacitance		20	pF	

A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = +5V \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{CY}	CLK Period	250	4000	ns	
t_{CL}	CLK Low Time	105	2000	ns	
t_{CH}	CLK High Time	105	2000	ns	
t_r	CLK Rise Time	0	30	ns	
t_f	CLK Fall Time	0	30	ns	
t_{AR}	A0, A1 Setup to $\overline{RD}\downarrow$	0		ns	
t_{AD}	A0, A1 to Data Output Delay		200	ns	$C_L = 150$ pF
t_{RA}	A0, A1 Hold after $\overline{RD}\uparrow$	0		ns	
t_{RD}	$\overline{RD}\downarrow$ to Data Output Delay		200	ns	$C_L = 150$ pF
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{DF}	Output Float Delay		120	ns	
t_{AW}	$\overline{CS}$, A0, A1 Setup to $\overline{WR}\downarrow$	0		ns	
t_{WA}	$\overline{CS}$, A0, A1 Hold after $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	150		ns	
t_{WD}	Data Hold after $\overline{WR}\uparrow$	0		ns	
t_{PI}	$\overline{IPI}$ Setup to $\overline{INTA}\downarrow$	0		ns	
t_{IP}	$\overline{IPI}$ Hold after $\overline{INTA}\uparrow$	10		ns	
t_{II}	$\overline{INTA}$ Pulse Width	250		ns	
t_{PIPO}	$\overline{IPI}\downarrow$ to $\overline{IPO}$ Delay		100	ns	
t_{ID}	$\overline{INTA}\downarrow$ to Data Output Delay		200	ns	
t_{CQ}	$\overline{RD}$ or $\overline{WR}$ to DRQ $\downarrow$		150	ns	
t_{RV}	Recovery Time Between Controls	300		ns	
t_{CW}	$\overline{CS}$, A0, A1 to RDY_A or RDY_B Delay		140	ns	
t_{DCY}	Data Clock Cycle	4.5		tcy	
t_{DCL}	Data Clock Low Time	180		ns	
t_{DCH}	Data Clock High Time	180		ns	
t_{TD}	$\overline{TxC}$ to TxD Delay (x1 Mode)		300	ns	
t_{DS}	RxD Setup to $\overline{RxC}\uparrow$	0		ns	
t_{DH}	RxD Hold after $\overline{RxC}\uparrow$	140		ns	
t_{ITD}	$\overline{TxC}$ to $\overline{INT}$ Delay	4	6	tcy	
t_{IRD}	RxC to $\overline{INT}$ Delay	7	10	tcy	
t_{PL}	$\overline{CTS}$, $\overline{CD}$, $\overline{SYNDET}$ Low Time	200		ns	
t_{PH}	$\overline{CTS}$, $\overline{CD}$, $\overline{SYNDET}$ High Time	200		ns	
t_{IPD}	External $\overline{INT}$ from $\overline{CTS}$, $\overline{CD}$, $\overline{SYNDET}$		500	ns	

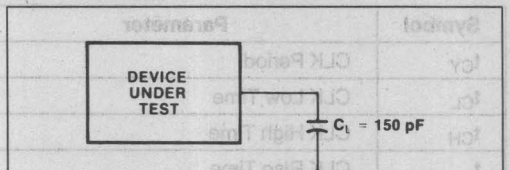
2

A.C. TESTING INPUT/OUTPUT WAVEFORM



A.C. Testing; Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

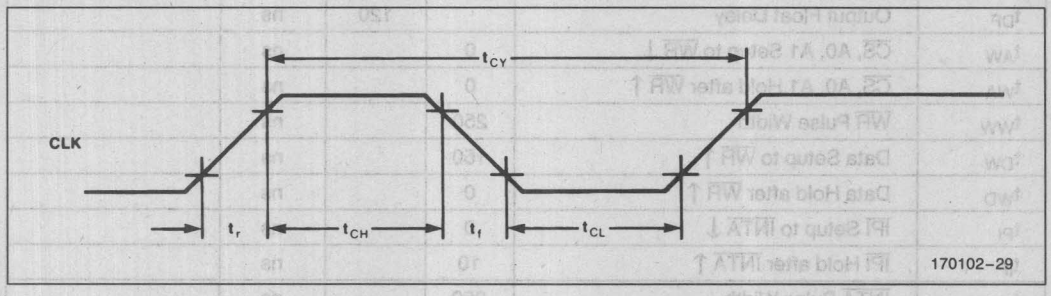
A.C. TESTING LOAD CIRCUIT



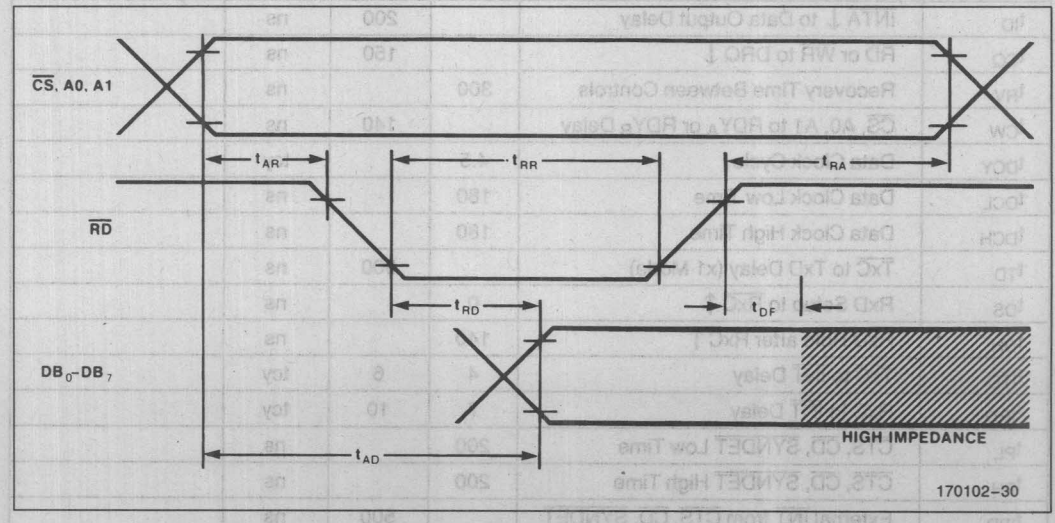
C_L = 100 pF
C_L Includes Jig Capacitance

WAVEFORMS

CLOCK CYCLE

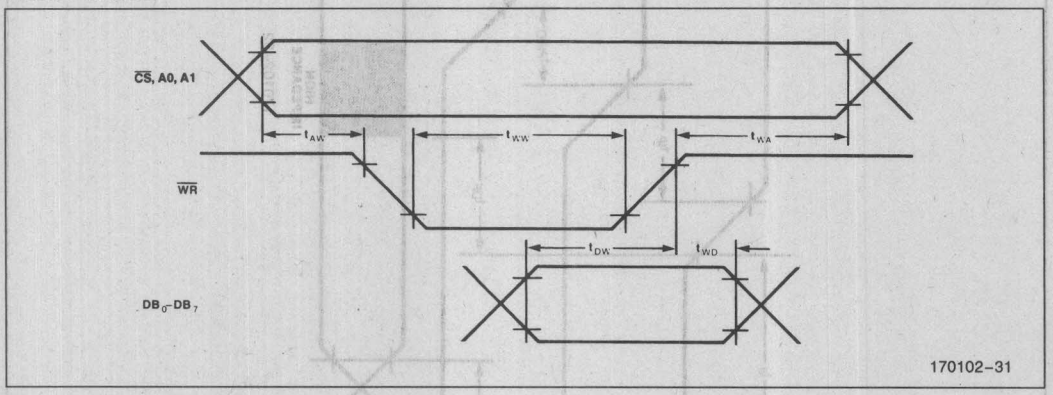


READ CYCLE

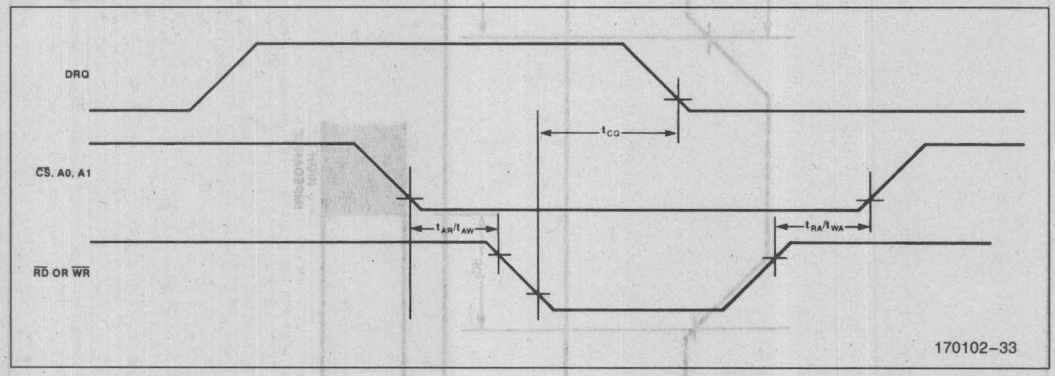


WAVEFORMS (Continued)

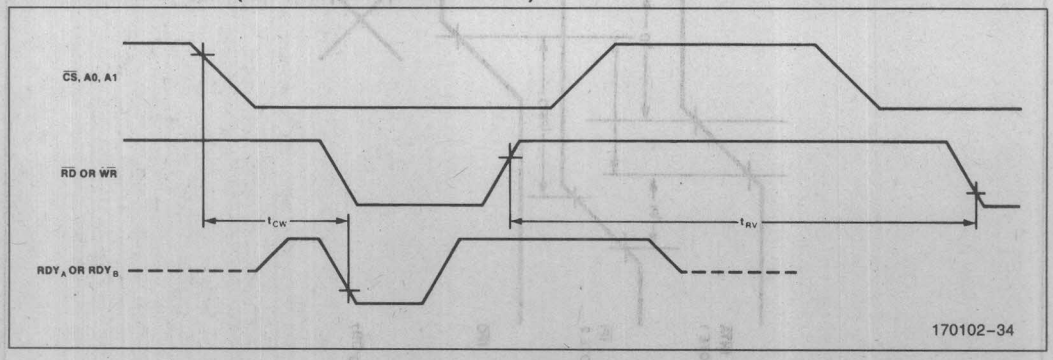
WRITE CYCLE



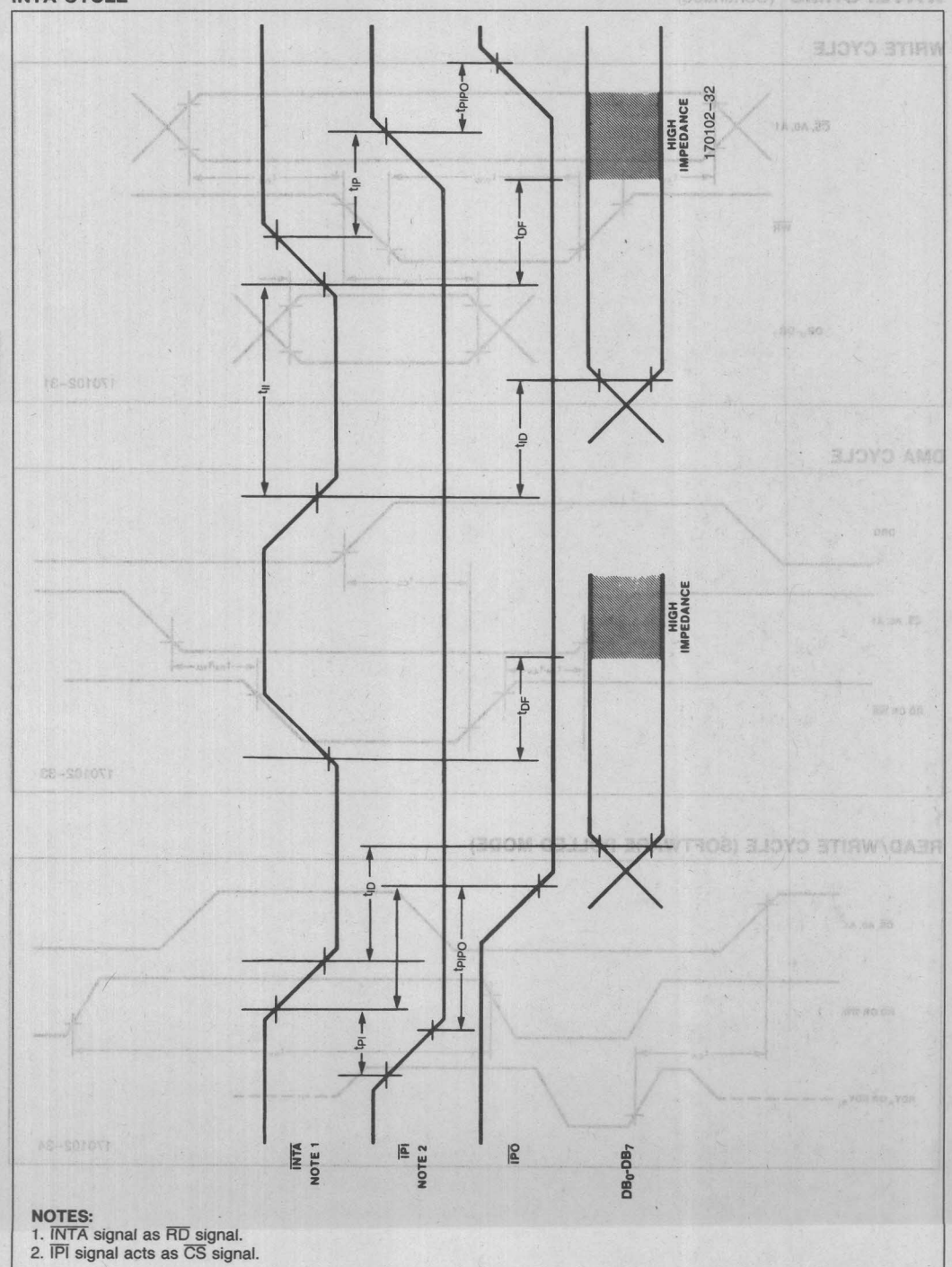
DMA CYCLE



READ/WRITE CYCLE (SOFTWARE POLLED MODE)



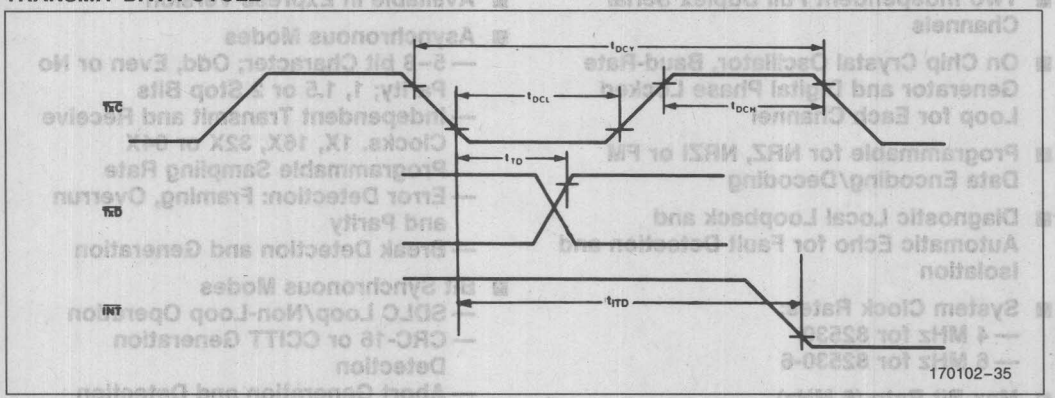
INTA CYCLE



NOTES:
1. $\overline{INTA}$ signal as $\overline{RD}$ signal.
2. $\overline{IPI}$ signal acts as $\overline{CS}$ signal.

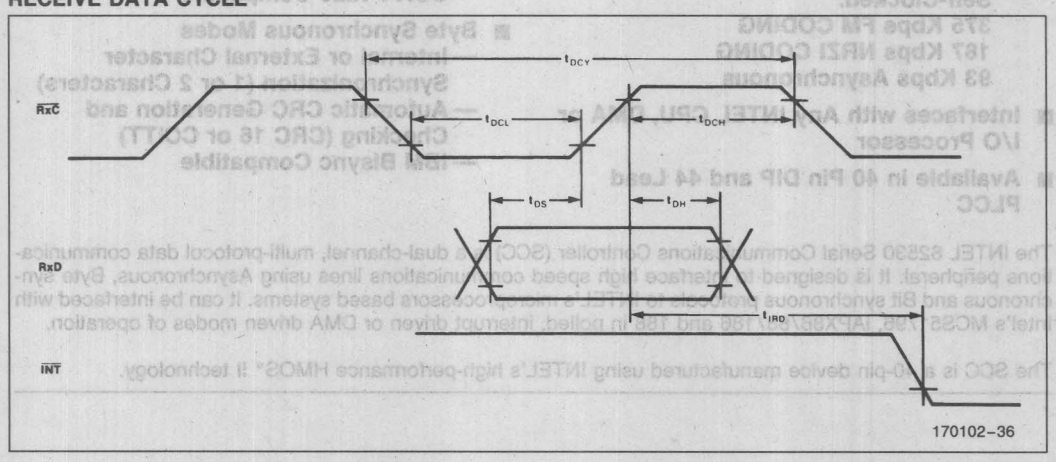
WAVEFORMS (Continued)

TRANSMIT DATA CYCLE



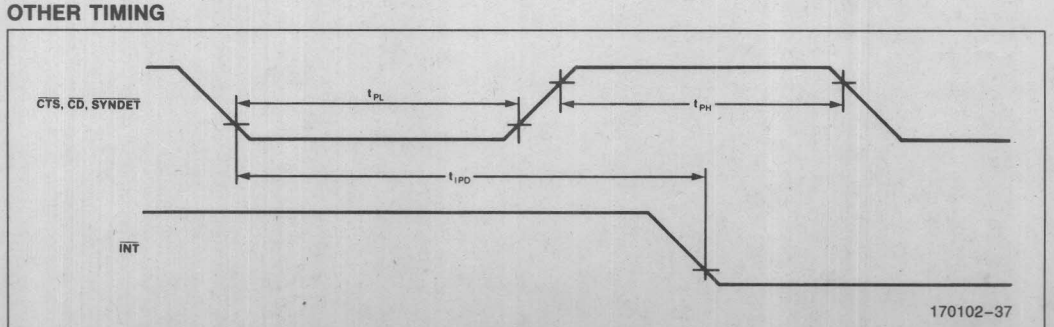
170102-35

RECEIVE DATA CYCLE



170102-36

OTHER TIMING



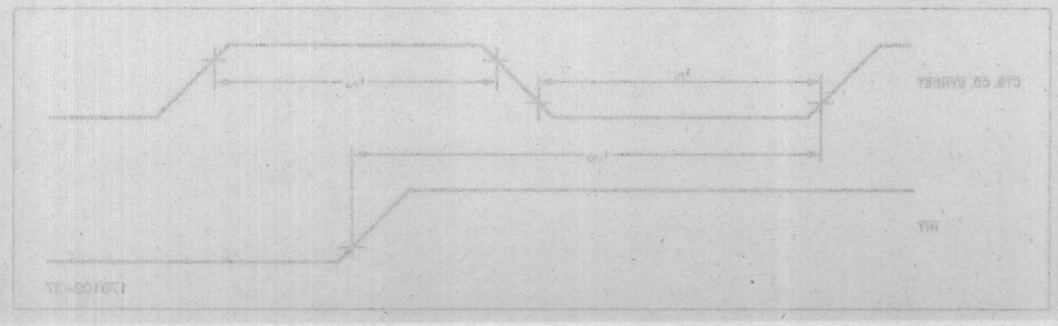
170102-37

82530/82530-6
SERIAL COMMUNICATIONS CONTROLLER (SCC)

- Two Independent Full Duplex Serial Channels
- On Chip Crystal Oscillator, Baud-Rate Generator and Digital Phase Locked Loop for Each Channel
- Programmable for NRZ, NRZI or FM Data Encoding/Decoding
- Diagnostic Local Loopback and Automatic Echo for Fault Detection and Isolation
- System Clock Rates:
  - 4 MHz for 82530
  - 6 MHz for 82530-6
- Max Bit Rate (6 MHz)
  - Externally Clocked: 1.5 Mbps
    Self-Clocked:
    375 Kbps FM CODING
    187 Kbps NRZI CODING
    93 Kbps Asynchronous
- Interfaces with Any INTEL CPU, DMA or I/O Processor
- Available in 40 Pin DIP and 44 Lead PLCC
- Available in Express Version
- Asynchronous Modes
  - 5–8 bit Character; Odd, Even or No Parity; 1, 1.5 or 2 Stop Bits
  - Independent Transmit and Receive Clocks. 1X, 16X, 32X or 64X Programmable Sampling Rate
  - Error Detection: Framing, Overrun and Parity
  - Break Detection and Generation
- Bit Synchronous Modes
  - SDLC Loop/Non-Loop Operation
  - CRC-16 or CCITT Generation Detection
  - Abort Generation and Detection
  - I-field Residue Handling
  - CCITT X.25 Compatible
- Byte Synchronous Modes
  - Internal or External Character Synchronization (1 or 2 Characters)
  - Automatic CRC Generation and Checking (CRC 16 or CCITT)
  - IBM Bisync Compatible

The INTEL 82530 Serial Communications Controller (SCC) is a dual-channel, multi-protocol data communications peripheral. It is designed to interface high speed communications lines using Asynchronous, Byte synchronous and Bit synchronous protocols to INTEL's microprocessors based systems. It can be interfaced with Intel's MCS51/96, iAPX86/88/186 and 188 in polled, interrupt driven or DMA driven modes of operation.

The SCC is a 40-pin device manufactured using INTEL's high-performance HMOS* II technology.



* HMOS is a patented process of Intel Corporation.

October 1987
Order Number: 230834-003

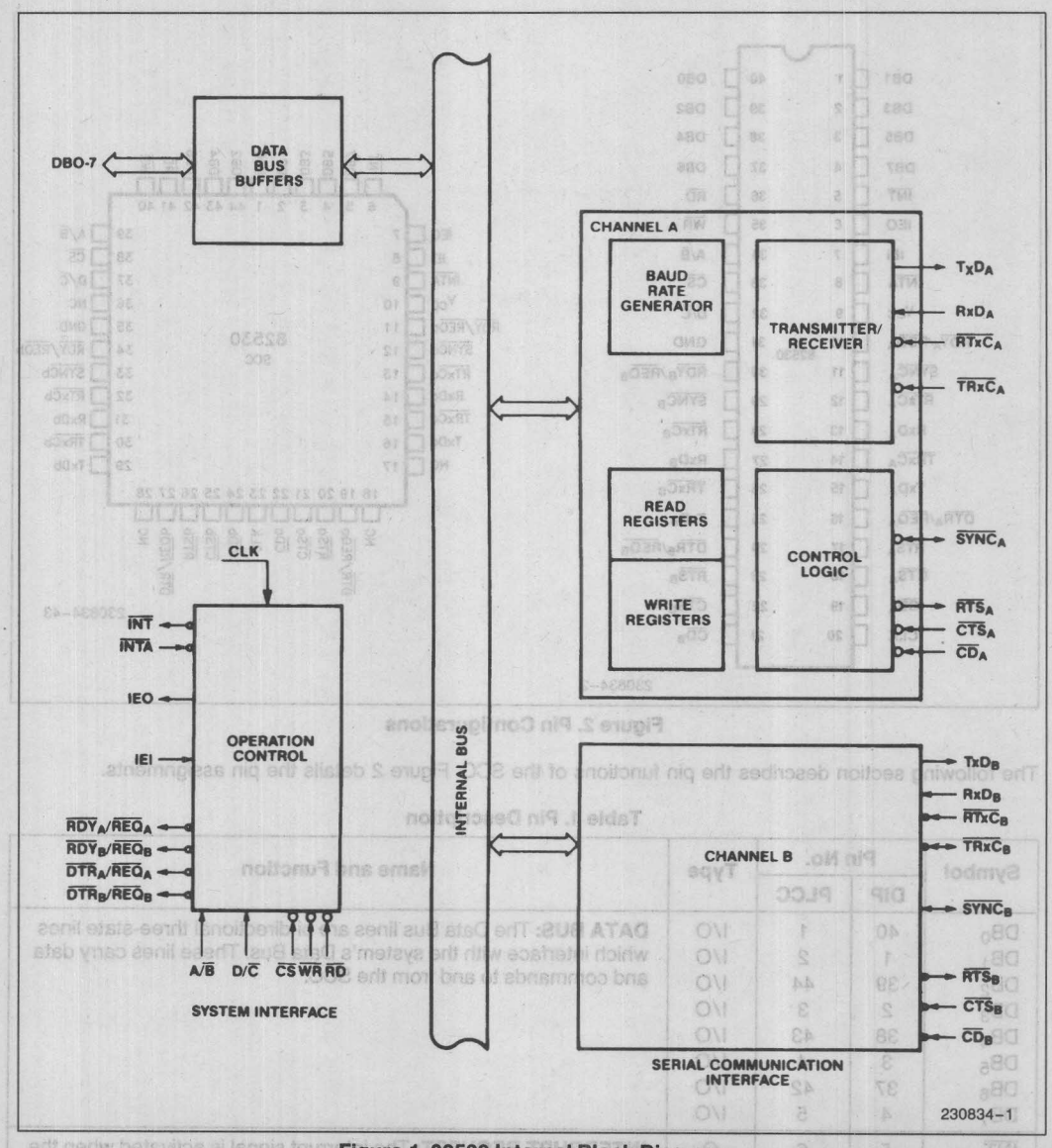


Figure 1. 82530 Internal Block Diagram

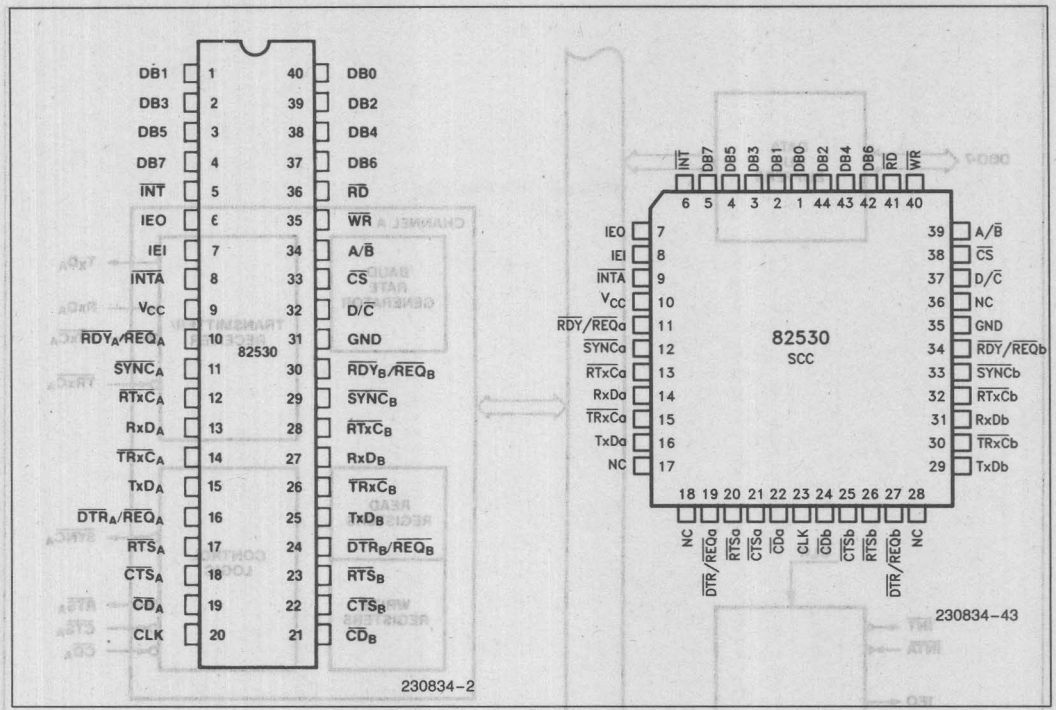


Figure 2. Pin Configurations

The following section describes the pin functions of the SCC. Figure 2 details the pin assignments.

Table 1. Pin Description

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
DB_0	40	1	I/O	**DATA BUS:** The Data Bus lines are bi-directional three-state lines which interface with the system's Data Bus. These lines carry data and commands to and from the SCC.
DB_1	1	2	I/O	
DB_2	39	44	I/O	
DB_3	2	3	I/O	
DB_4	38	43	I/O	
DB_5	3	4	I/O	
DB_6	37	42	I/O	
DB_7	4	5	I/O	
$\overline{INT}$	5	6	O	**INTERRUPT REQUEST:** The interrupt signal is activated when the SCC requests an interrupt. It is an open drain output.
IEO	6	7	O	**INTERRUPT ENABLE OUT:** IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.
IEI	7	8	I	**INTERRUPT ENABLE IN:** IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

Table 1. Pin Description (Continued)

Symbol	Pin No. DIP	Pin No. PLCC	Type	Name and Function
$\overline{INTA}$	8	9	I	**INTERRUPT ACKNOWLEDGE:** This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{RD}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTA}$ is latched by the rising edge of CLK.
V_{CC}	9	10		**POWER:** +5V Power supply.
$\overline{RDY}_A/\overline{REQ}_A$ $\overline{RDY}_B/\overline{REQ}_B$	10 30	11 34	O O	**READY/REQUEST:** (output, open-drain when programmed for a Ready function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Ready lines to synchronize the CPU to the SCC data rate. The reset state is Ready.
$\overline{SYNC}_A$ $\overline{SYNC}_B$	11 29	12 33	I/O I/O	**SYNCHRONIZATION:** These pins can act either as inputs, outputs or part of the crystal oscillator circuit. In the Asynchronous receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{CD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 9) but have no other function. In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven LOW two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$. In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of characters boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.
$\overline{RTxC}_A$ $\overline{RTxC}_B$	12 28	13 32	I I	**RECEIVE/TRANSMIT CLOCKS:** These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase Locked Loop. These pins can be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.
RxD_A RxD_B	13 27	14 31	I I	**RECEIVE DATA:** These lines receive serial data at standard TTL levels.
$\overline{TRxC}_A$ $\overline{TRxC}_B$	14 26	15 30	I/O I/O	**TRANSMIT/RECEIVE CLOCKS:** These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.
TxD_A TxD_B	15 25	16 29	O O	**TRANSMIT DATA:** These output signals transmit serial data at standard TTL levels
$\overline{DTR}_A/\overline{REQ}_A$ $\overline{DTR}_B/\overline{REQ}_B$	16 24	19 27	O O	**DATA TERMINAL READY/REQUEST:** These outputs follow the state programmed into the DTR bit. They can also be used as general purpose outputs or as Request lines for a DMA controller.

Table 1. Pin Description (Continued)

Symbol	Pin No.		Type	Name and Function
	DIP	PLCC		
$\overline{RTS}_A$ $\overline{RTS}_B$	17 23	20 26	O O	**REQUEST TO SEND:** When the Request to Send (RTS) bit in Write Register 5 is set (Figure 10), the $\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.
$\overline{CTS}_A$ $\overline{CTS}_B$	18 22	21 25	I I	**CLEAR TO SEND:** If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.
$\overline{CD}_A$ $\overline{CD}_B$	19 21	22 24	I I	**CARRIER DETECT:** These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.
CLK	20	23	I	**CLOCK:** This is the system SCC clock used to synchronize internal signals.
GND	31	35		**GROUND**
D/$\overline{C}$	32	37	I	**DATA/COMMAND SELECT:** This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.
$\overline{CS}$	33	38	I	**CHIP SELECT:** This signal selects the SCC for a read or write operation.
A/$\overline{B}$	34	39	I	**CHANNEL A/CHANNEL B SELECT:** This signal selects the channel in which the read or write operation occurs.
$\overline{WR}$	35	40	I	**WRITE:** When the SCC is selected this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.
$\overline{RD}$	36	41	I	**READ:** This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

GENERAL DESCRIPTION

The Intel 82350 Serial Communications Controller (SCC) is a dual-channel, multi-protocol data communications peripheral. The SCC functions as a serial-to-parallel, parallel-to-serial convertor/controller. The SCC can be software-configured to satisfy a wide range of serial communications applications. The device contains sophisticated internal functions including on-chip baud rate generators, digital phase locked loops, various data encoding and decoding schemes, and crystal oscillators that reduce the need for external logic.

In addition, diagnostic capabilities—automatic echo and local loopback—allow the user to detect and isolate a failure in the network. They greatly improve the reliability and fault isolation of the system.

The SCC handles Asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (Terminal, Personal Computer, Peripherals, Industrial Controller, Telecommunication sytem, etc.).

The 82530 can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem control in both channels. In applications where these controls are not needed, the modem control can be used for general purpose I/O.

The Intel 82530 is designed to support Intel's MCS51/96, iAPX 86/88 and iAPX 186/188 families.

ARCHITECTURE

The 82530 internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a non-multiplexed CPU bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface modems or other external devices.

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two synchronous character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write-only Master Interrupt Control register and three read registers; one containing the vector with status information (Channel B only), one containing the vector without status (A only), and one containing the Interrupt Pending bits (A only).

The registers for each channel are designated as follows:

WR0–WR15—Write Registers 0 through 15.

RR0–RR3, RR10, RR12, RR13, RR15—Read Registers 0 through 3, 10, 12, 13, 15.

Table 2 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

DATA PATH

The transmit and receive data path illustrated in Figure 3 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in asynchronous modes also determines the data path).

The transmitter has an 8-bit transmit data buffer register loaded from the internal data bus and a 20-bit transmit shift register that can be loaded either from the sync-character registers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

Table 2. Read and Write Register Functions

READ REGISTER FUNCTIONS		WRITE REGISTER FUNCTIONS	
RR0	Transmit/Receive buffer status and External status	**WR0**	CRC initialize, initialization commands for the various modes, shift right/shift left command
RR1	Special Receive Condition status	**WR1**	Transmit/Receive interrupt and data transfer mode definition
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt (Channel A only)	**WR2**	Interrupt vector (accessed through either channel)
RR3	Interrupt Pending bits (Channel A only)	**WR3**	Receive parameters and control
RR8	Receive buffer	**WR4**	Transmit/Receive miscellaneous parameters and modes
RR10	Miscellaneous status	**WR5**	Transmit parameters and controls
RR12	Lower byte of baud rate generator time constant	**WR6**	Sync characters or SDLC address field
RR13	Upper byte of baud rate generator time constant	**WR7**	Sync character or SDLC flag
RR15	External/Status interrupt information	**WR8**	Transmit buffer
		WR9	Master interrupt control and reset (accessed through either channel)
		WR10	Miscellaneous transmitter/receiver control bits
		WR11	Clock Mode control
		WR12	Lower Byte of baud rate generator time constant
		WR13	Upper Byte of baud rate generator time constant
		WR14	Miscellaneous control bits
		WR15	External/Status interrupt control

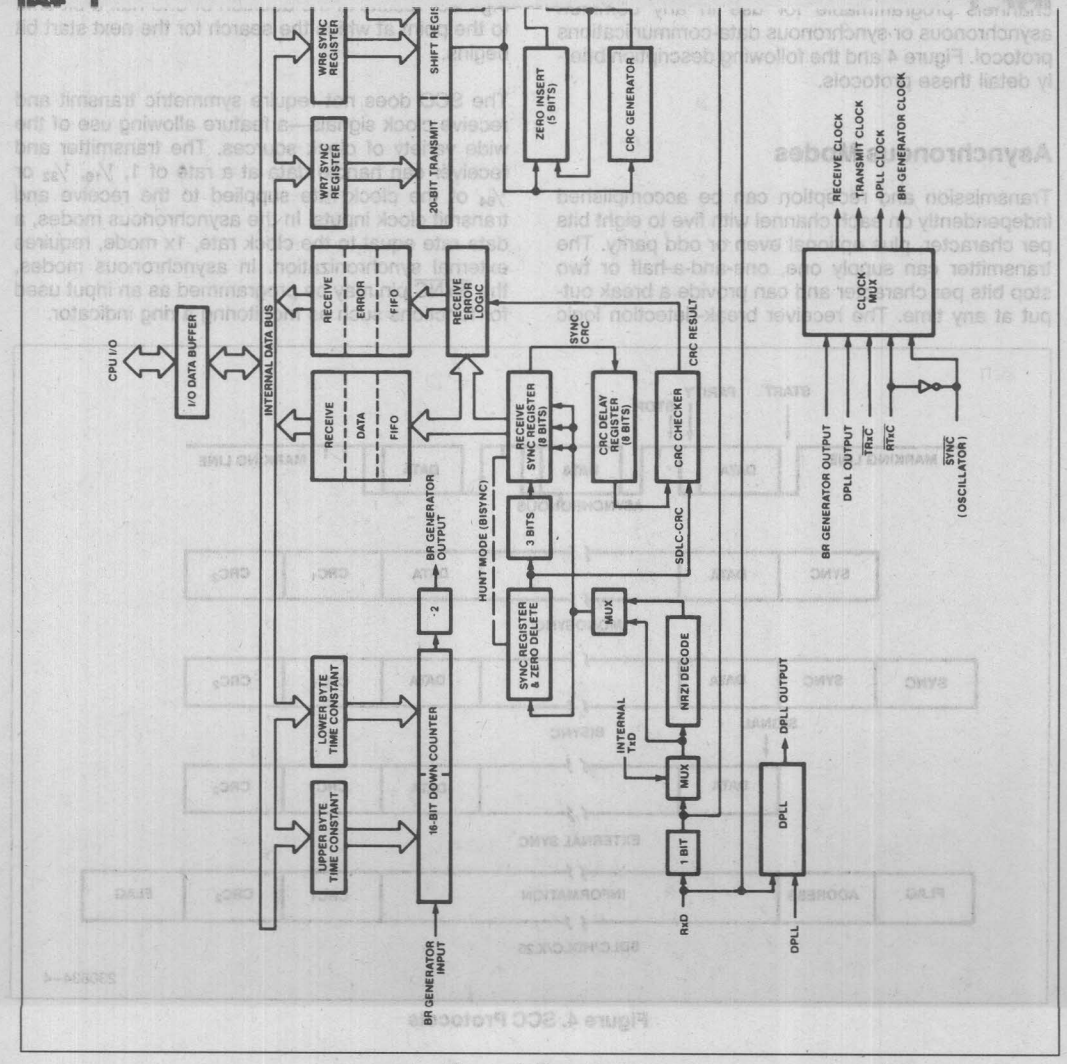


Figure 3. Data Path

FUNCTIONAL DESCRIPTION

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, it interacts with the CPU and provides vectored interrupts and handshaking signals.

DATA COMMUNICATIONS CAPABILITIES

The SCC provides two independent full-duplex channels programmable for use in any common asynchronous or synchronous data-communications protocol. Figure 4 and the following description briefly detail these protocols.

Asynchronous Modes

Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxD_A or RxD_B). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In the asynchronous modes, a data rate equal to the clock rate, 1x mode, requires external synchronization. In asynchronous modes, the $\overline{SYNC}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

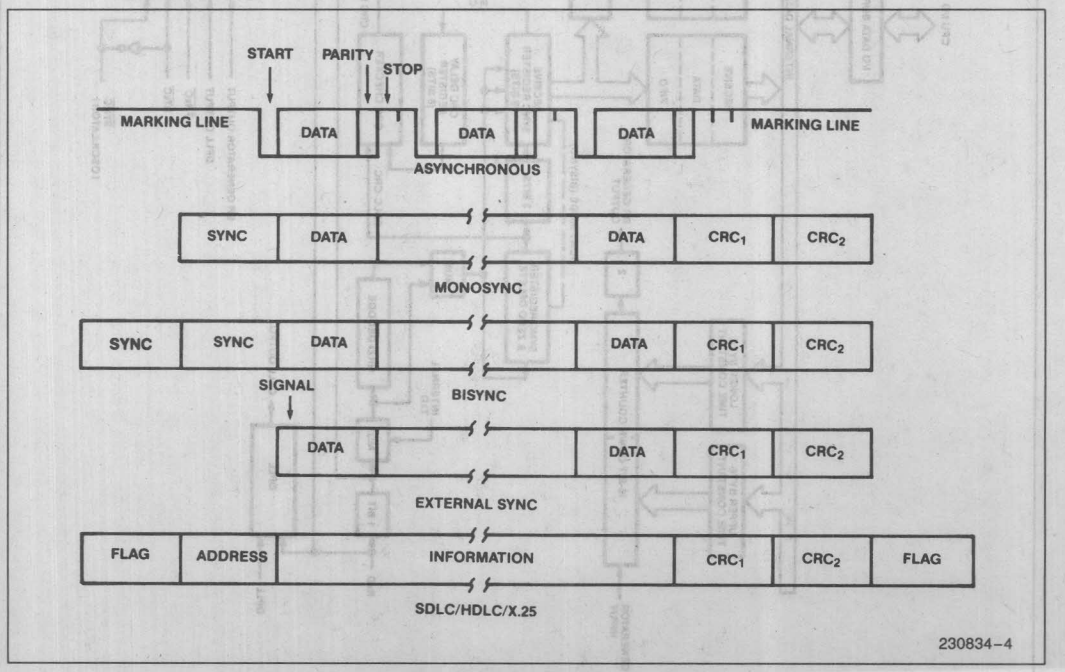


Figure 4. SCC Protocols

Synchronous Modes

The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous-byte-oriented protocols can be handled in several modes allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit or 16-bit synchronous pattern (Bisync), or with an external synchronous signal. Leading synchronous characters can be removed without interrupting the CPU.

5- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 5.

CRC checking for Synchronous byte-oriented mode is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high-speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC mode and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s. The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1X mode. The parity options available in asynchronous modes are available in synchronous modes.

The SCC can be conveniently used under DMA control to provide high-speed reception or transmission.

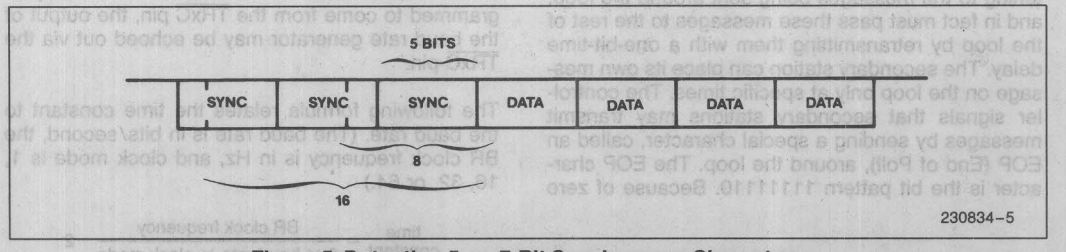


Figure 5. Detecting 5- or 7-Bit Synchronous Characters

In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via DMA.

SDLC LOOP MODE

The SCC supports SDLC Loop mode in addition to normal SDLC. In a loop topology, there is a primary controller station that manages the message traffic flow and any number of secondary stations. In Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 6).

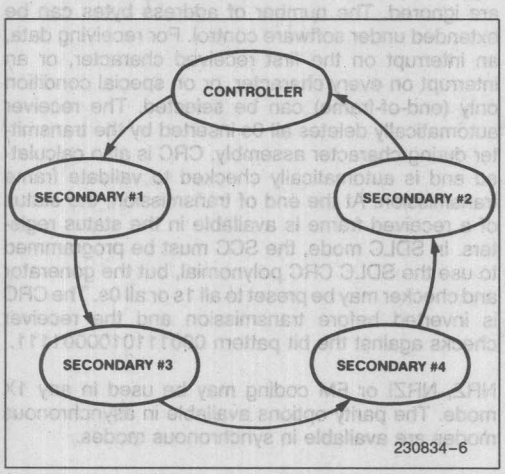


Figure 6. An SDLC Loop

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary one of the EOP to a zero before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

BAUD RATE GENERATORS

Each channel in the SCC contains a programmable Baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching zero, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the digital phase-locked loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate. (The baud rate is in bits/second, the BR clock frequency is in Hz, and clock mode is 1, 16, 32, or 64.)

$$\text{time constant} = \frac{\text{BR clock frequency}}{2 \times \text{baud rate} \times \text{clock mode}} - 2$$

Table 3. Time Constant Values for Standard Baud Rates at BR Clock = 3.9936 MHz

Rate (BAUD)	Time Constant (decimal notation)	Error
19200	102	—
9600	206	—
7200	275	0.12%
4800	414	—
3600	553	0.06%
2400	830	—
2000	996	0.04%
1800	1107	0.03%
1200	1662	—
600	3326	—
300	6654	—
150	13310	—
134.5	14844	0.0007%
110	18151	0.0015%
75	26622	—
50	39934	—

DIGITAL PHASE LOCKED LOOP

The SCC contains a digital phase locked-loop (DPLL) to recover clock information from a datastream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the datastream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI coding, the DPLL counts the 32X clock to create nominal bit times. As the 32X clock is counted, the DPLL is searching the incoming datastream for edges (either 1/0 or 0/1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 1 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the datastream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the $^{15}/_{16}$ counting transition.

The 32X clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

DATA ENCODING

The SCC may be programmed to encode and decode the serial data in four different ways (Figure 7).

In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, as 1 is represented by no change in level and a 0 is represented by a change in level. In FM_1 (more properly, bi-phase mark) a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM_0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0/1 the bit is a 0. If the transition is 1/0 the bit is a 1.

AUTO ECHO AND LOCAL LOOPBACK

The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in asynchronous modes, but works in synchronous and SDLC modes as well. In Auto Echo mode TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the datastream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{READY}}$/$\overline{\text{REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode, TxD is RxD just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in asynchronous, synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

SERIAL BIT RATE

To run the 82530 (4 MHz/6 MHz) at 1/1.5 Mbps the receive and transmit clocks must be externally generated and synchronized to the system clock. If the serial clocks ($\overline{\text{RTxC}}$ and $\overline{\text{TRxC}}$) and the system clock (CLK) are asynchronous, the maximum bit rate is 880 Kbps/1.3 Mbps. For self-clocked operation, i.e using the on chip DPLL, the maximum bit rate is 125/187 Kbps if NRZI coding is used and 250/375 Kbps if FM coding is used.

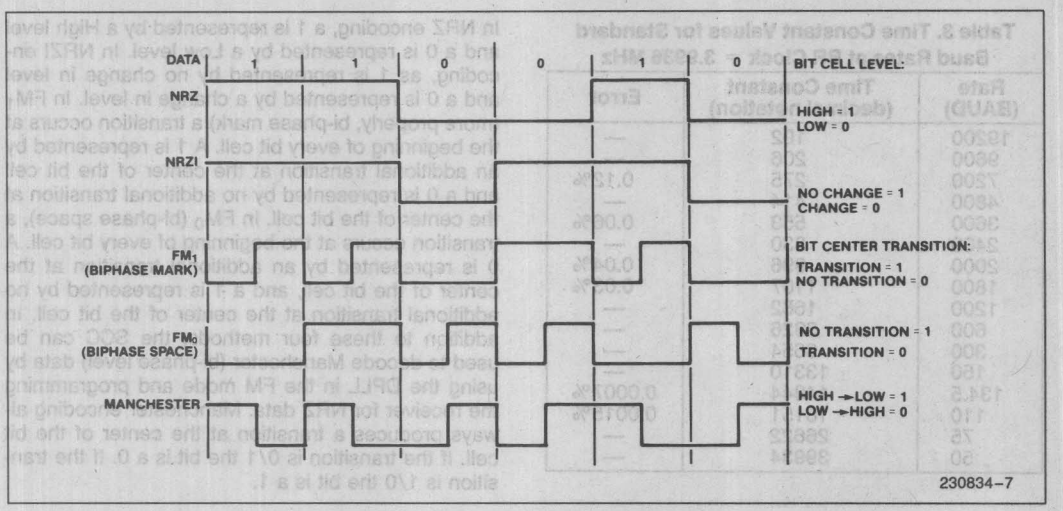


Figure 7. Data Encoding Methods

Table 4. Maximum Bit Rates

Mode	System Clock	System Clock/ Serial Clock	Serial Bit Rate	Conditions
Serial clocks generated externally	4 MHz	4	1 Mbps	Serial clocks synchronized with system clock. Refer to parameter #3 and 10 in general timings.
	6 MHz	4	1.5 Mbps	Serial clocks synchronized with system clock. Refer to parameter #3 and #10 in general timings.
	4 MHz	4.5	880 Kbps	Serial clocks and system clock asynchronous.
	6 MHz	4.5	1.3 Mbps	Serial clocks and system clock asynchronous
Self-clocked operation				
NRZI	4 MHz	32	125 Kbps	
	6 MHz	32	187 Kbps	
FM	4 MHz	16	250 Kbps	
	6 MHz	16	375 Kbps	
ASYNC	4 MHz	16	62.5 Kbps	
	6 MHz	16	93.75 Kbps	

I/O INTERFACE CAPABILITIES

The SCC offers the choice of Polling, Interrupt (vectored or nonvectored) and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

POLLING

All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

INTERRUPTS

When a SCC responds to an Interrupt Acknowledge signal ($\overline{INTA}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 9 and 10).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bits is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write-only.

The other two bits are related to the interrupt priority chain (Figure 8). As a peripheral, the SCC may request an interrupt only when no higher-priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTA}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive and External/Status interrupts. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive condition.
- Interrupt on all Receive Characters or Special Receive condition.
- Interrupt on Special Receive condition only.

Interrupt-on-First-Character or Special-Condition and Interrupt-on-Special-Condition-Only are typically used with the Block Transfer mode. A Special-Receive-Condition is one of the following: receiver overrun, framing error in Asynchronous mode, End-of-Frame in SDLC mode and, optionally, a parity error. The Special-Receive-Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector

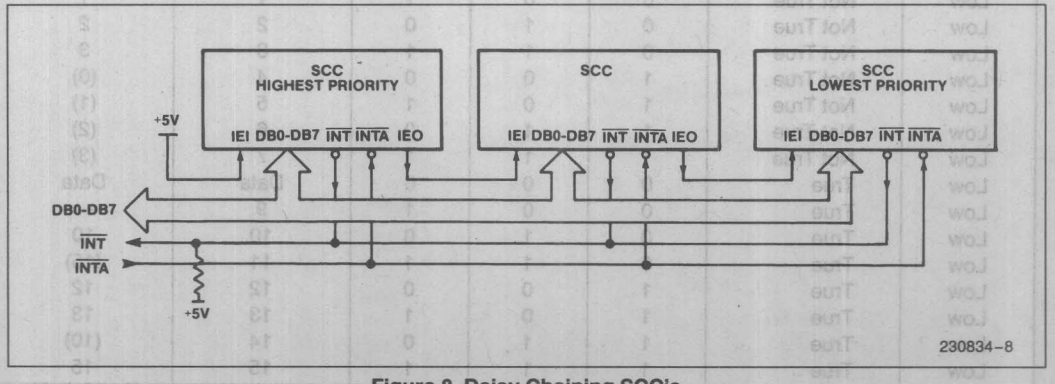


Figure 8. Daisy Chaining SCC's

during the Interrupt-Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{CD}$, and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA BLOCK TRANSFER

The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{READY}$/$\overline{REQUEST}$ output in conjunction with the $\overline{READY}$/$\overline{REQUEST}$ bits in WR1. The $\overline{READY}$/$\overline{REQUEST}$ output can be defined under software control as a $\overline{READY}$ line in the CPU Block Transfer mode (WR1; D6 = 0) or as a request line in the DMA Block Transfer mode (WR1; D6 = 1). To a DMA controller, the SCC $\overline{REQUEST}$ output indicates that the SCC is ready to transfer data to or from memory. To the CPU, The $\overline{READY}$ line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{DTR}$/$\overline{REQUEST}$ line allows full-duplex operation under DMA control.

PROGRAMMING

Each channel has fifteen Write registers that are individually programmed from the system bus to configure the functional personality of each channel. Each channel also has eight Read registers from which the system can read Status, Baud rate, or Interrupt information.

Only the four data registers (Read, Write for channels A and B) are directly selected by a High on the D/$\overline{C}$ input and the appropriate levels on the $\overline{RD}$, $\overline{WR}$ and A/$\overline{B}$ pins. All other registers are addressed indirectly by the content of Write Register 0 in conjunction with a Low on the D/$\overline{C}$ input and the appropriate levels on the $\overline{RD}$, $\overline{WR}$ and A/$\overline{B}$ pins. If bit 3 in WW0 is 1 and bits 4 and 5 are 0 then bits 0, 1, 2 address the higher registers 8 through 15. If bits 3, 4, 5 contain a different code, bits 0, 1, 2 address the lower registers 0 through 7 as shown on Table 5.

Writing to or reading from any register except RR0, WR0 and the Data Registers thus involves two operations.

Table 5. Register Addressing

D/C "Point High" Code in WR0		D2 in WR0	D1 in WR0	D0 in WR0	Write Register	Read Register
High	Either Way	X	X	X	Data	Data
Low	Not True	0	0	0	0	0
Low	Not True	0	0	1	1	1
Low	Not True	0	1	0	2	2
Low	Not True	0	1	1	3	3
Low	Not True	1	0	0	4	(0)
Low	Not True	1	0	1	5	(1)
Low	Not True	1	1	0	6	(2)
Low	Not True	1	1	1	7	(3)
Low	True	0	0	0	Data	Data
Low	True	0	0	1	9	—
Low	True	0	1	0	10	10
Low	True	0	1	1	11	(15)
Low	True	1	0	0	12	12
Low	True	1	0	1	13	13
Low	True	1	1	0	14	(10)
Low	True	1	1	1	15	15

First write the appropriate code into WR0, then follow this by a write or read operation on the register thus specified. Bits 0 through 4 in WW0 are automatically cleared after this operation, so that WW0 then points to WR0 or RR0 again.

Channel A/Channel B selection is made by the A/$\overline{B}$ input (High = A, Low = B)

The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

READ REGISTERS

The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to determine the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 9 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring: e.g. when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

WRITE REGISTERS

The SCC contains 15 write registers (16 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 10 shows the format of each write register.

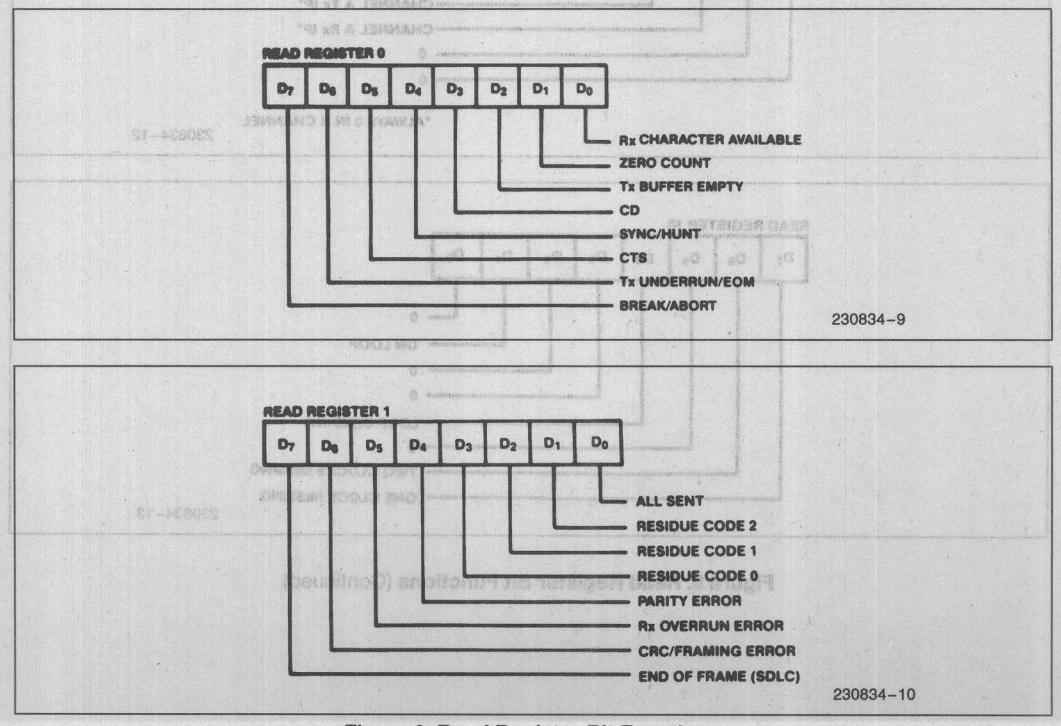
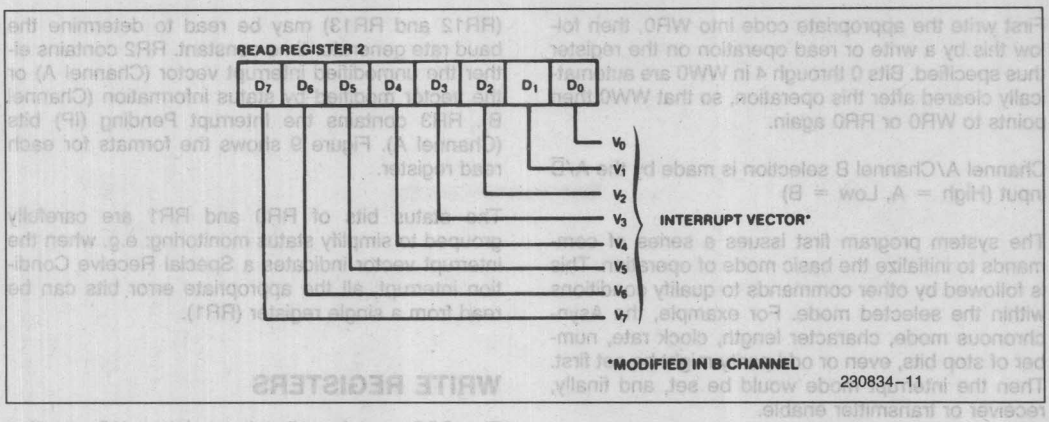
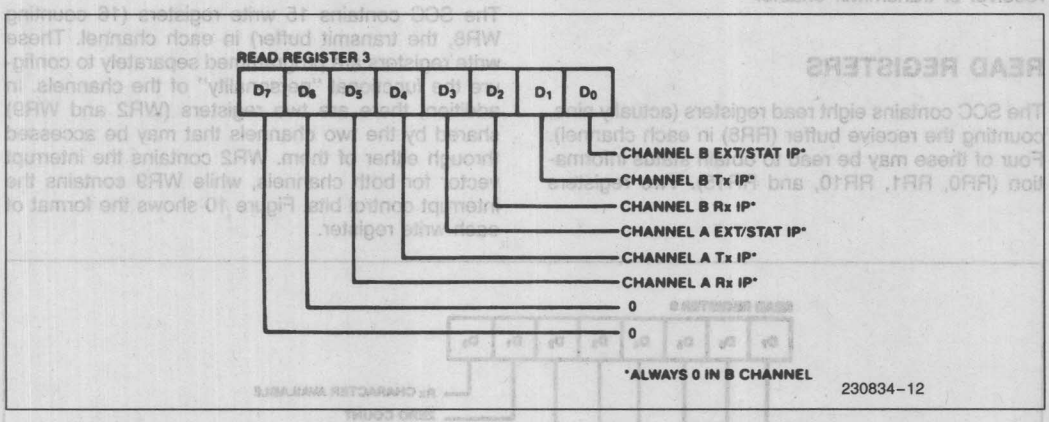


Figure 9. Read Register Bit Functions





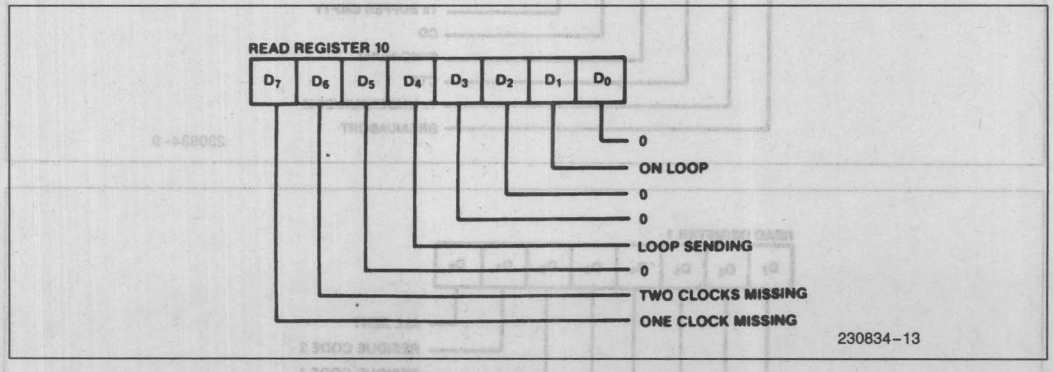
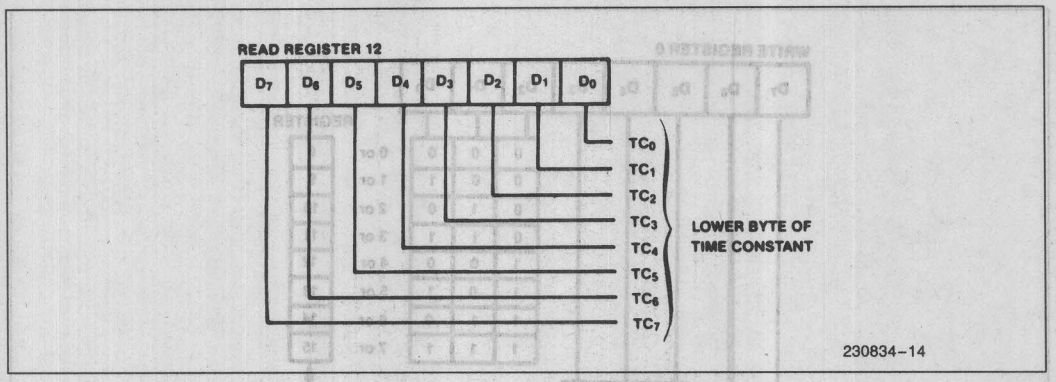
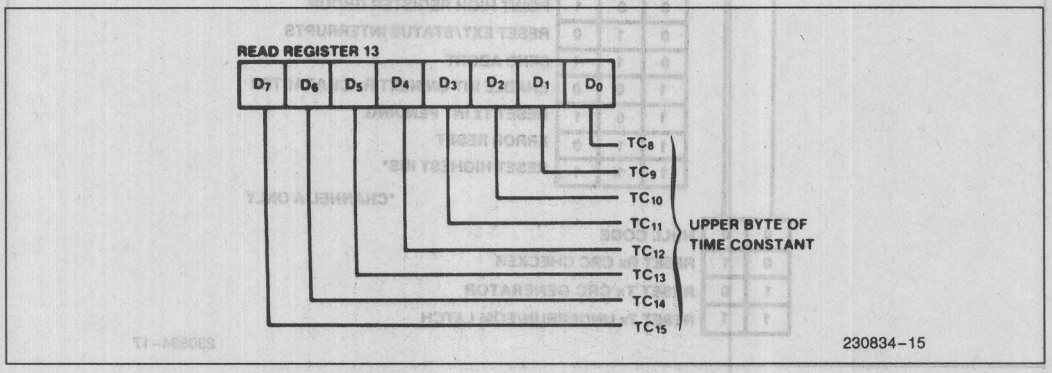


Figure 9. Read Register Bit Functions (Continued)





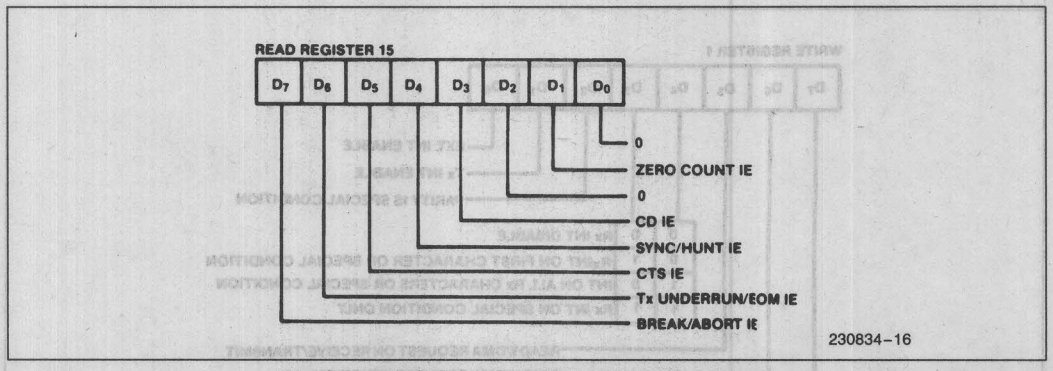
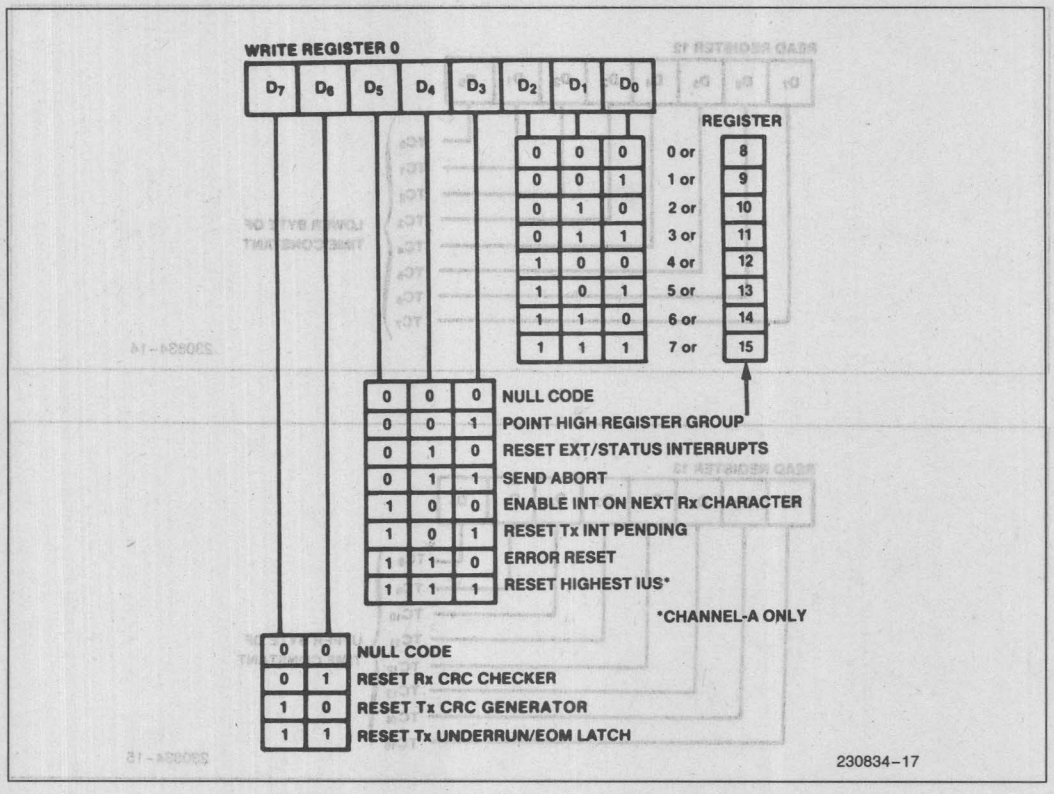


Figure 9. Read Register Bit Functions (Continued)



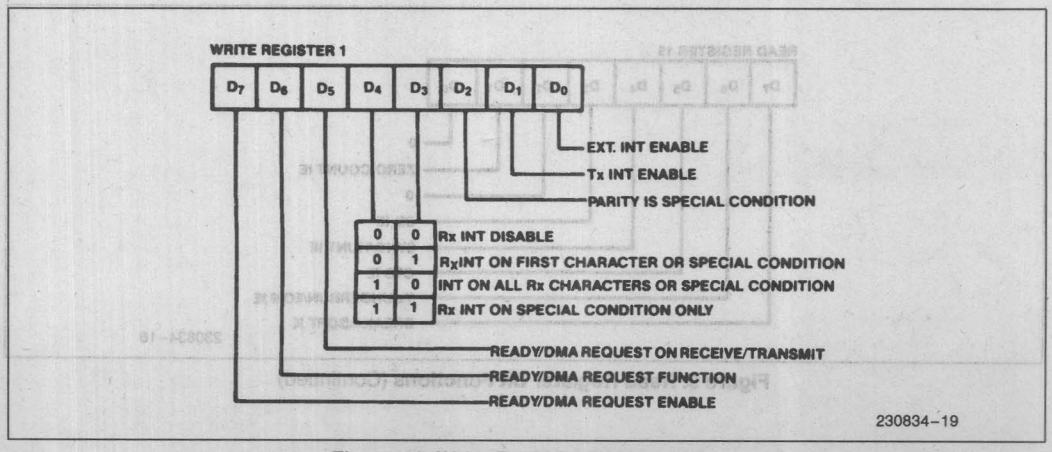
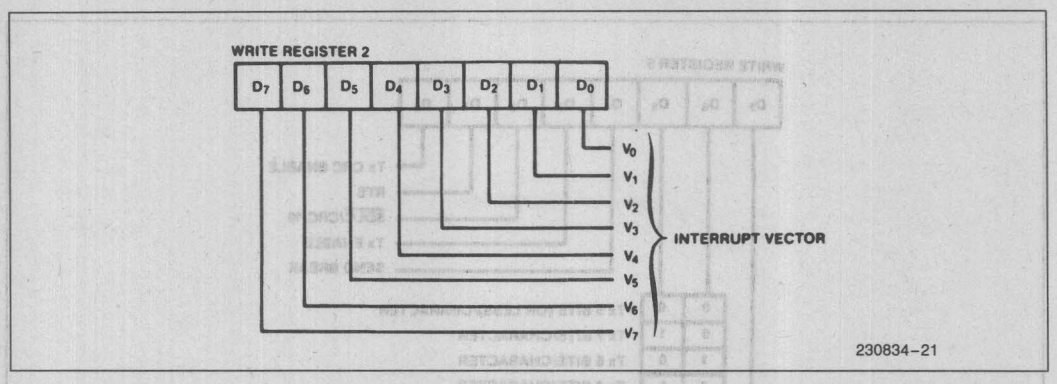
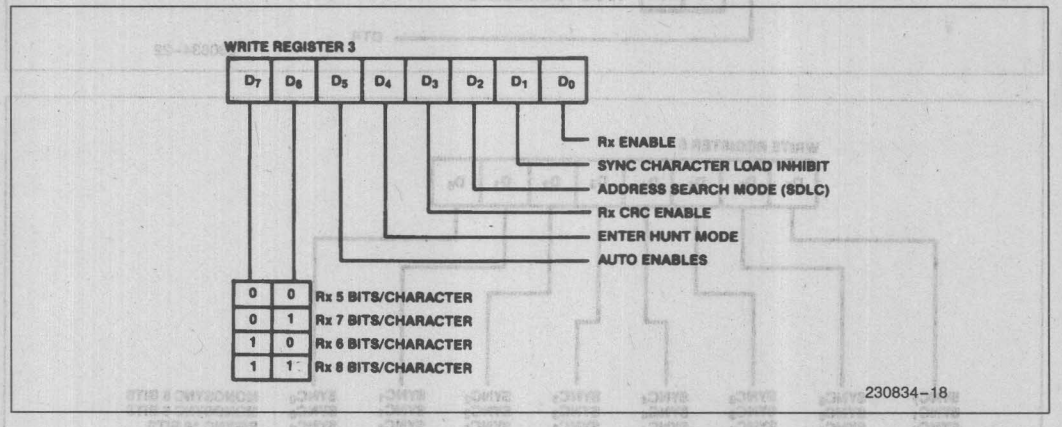


Figure 10. Write Register Bit Functions





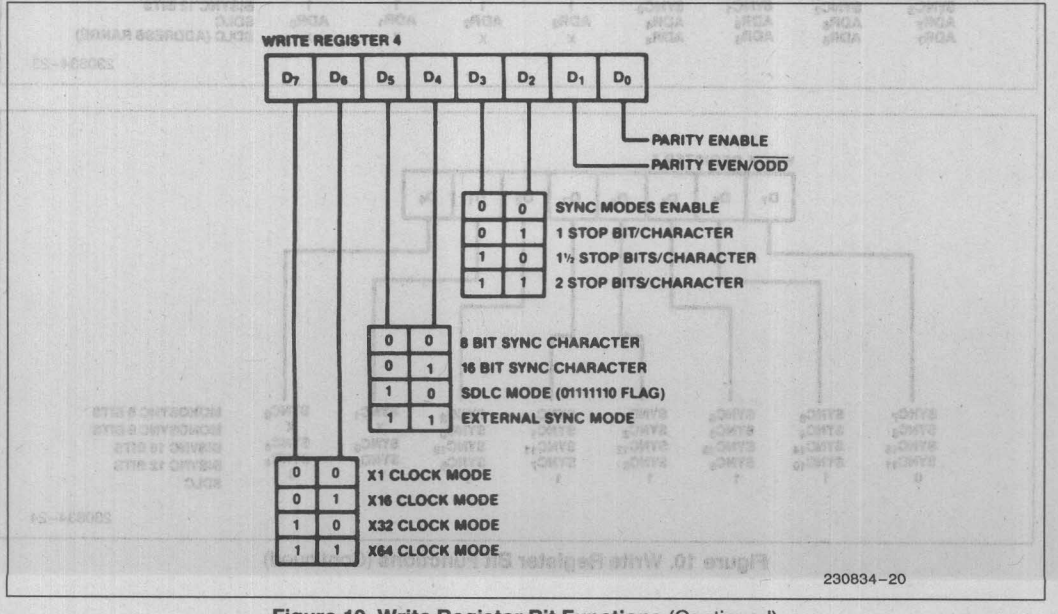
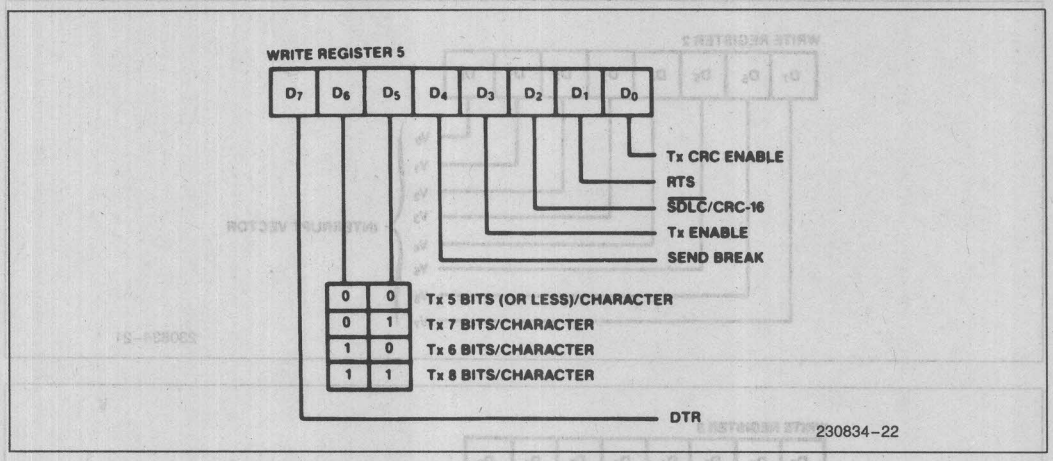
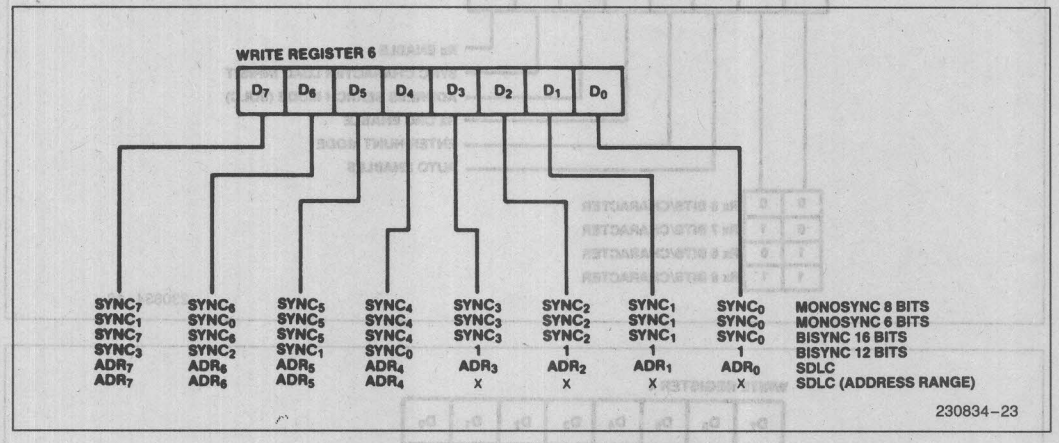


Figure 10. Write Register Bit Functions (Continued)





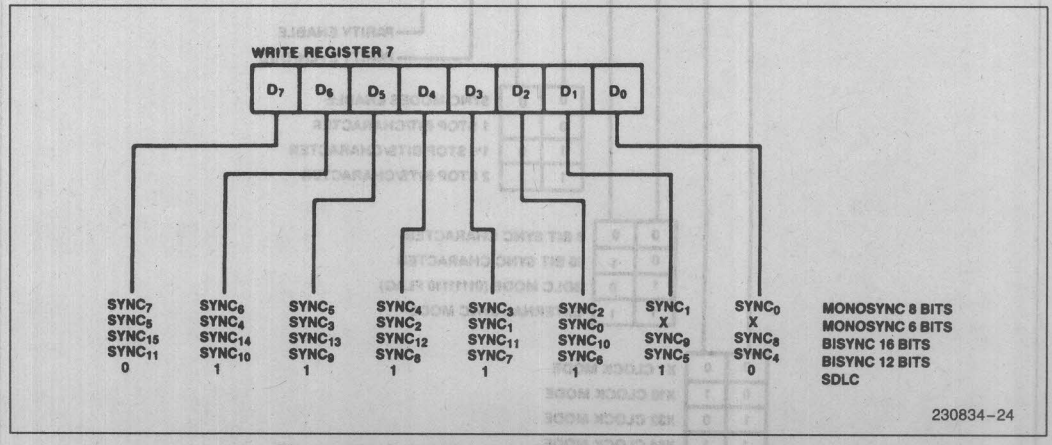
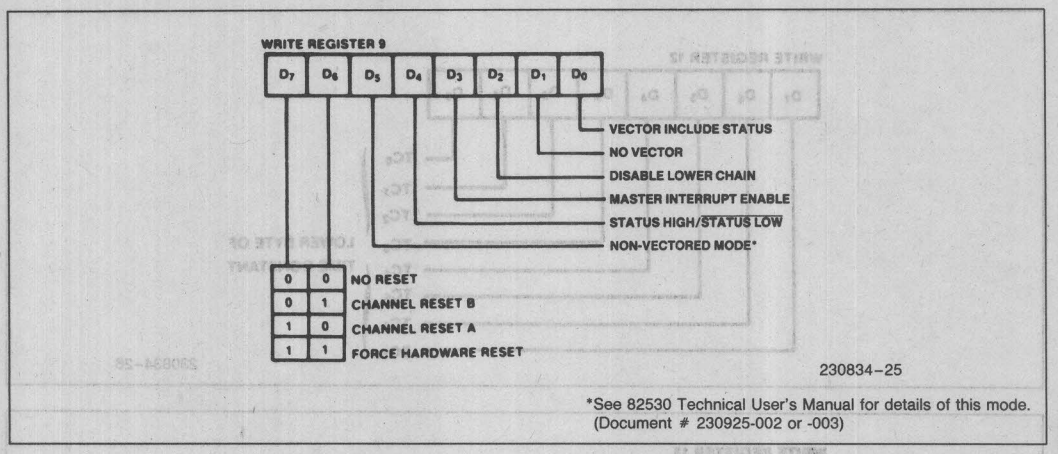
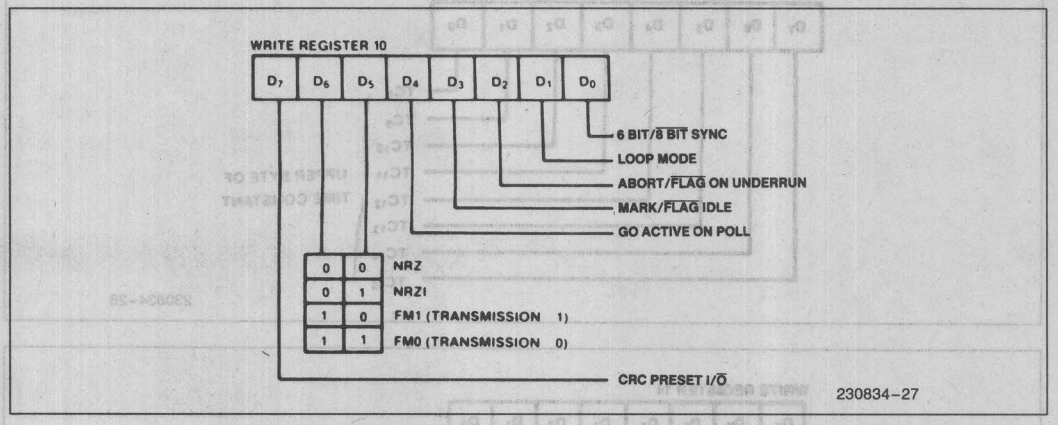


Figure 10. Write Register Bit Functions (Continued)



*See 82530 Technical User's Manual for details of this mode. (Document # 230925-002 or -003)



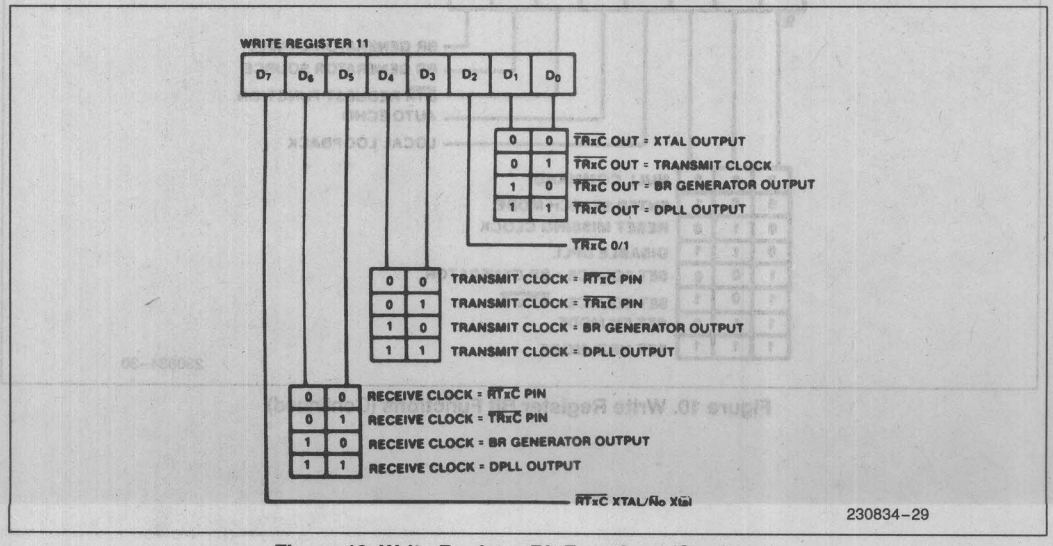
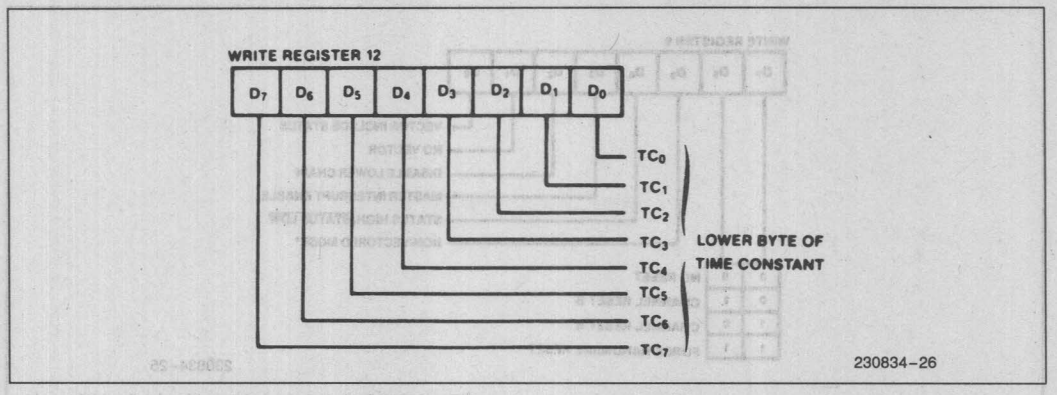
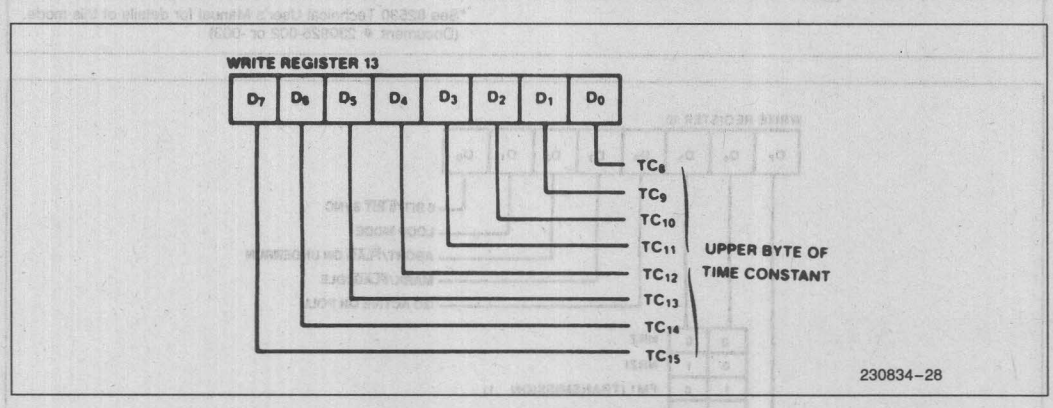


Figure 10. Write Register Bit Functions (Continued)





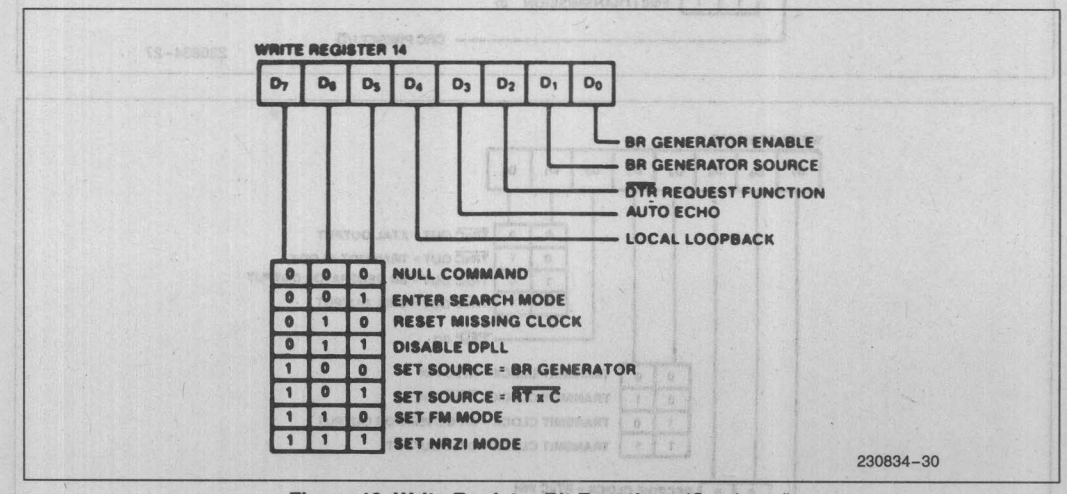


Figure 10. Write Register Bit Functions (Continued)

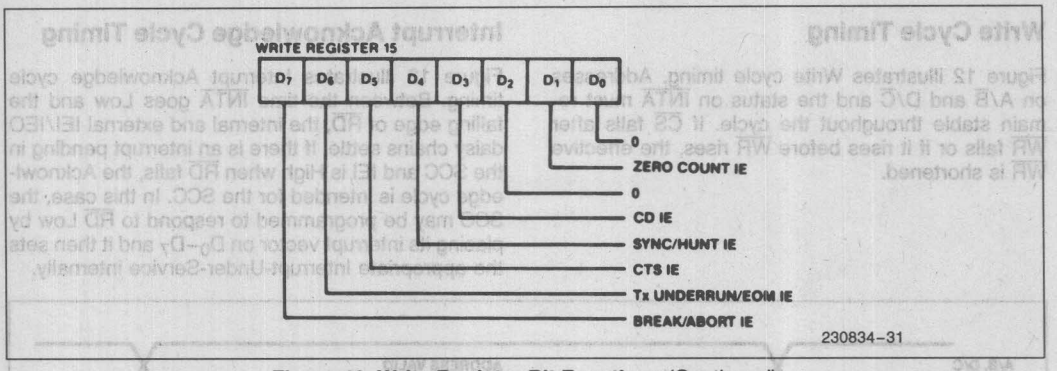


Figure 10. Write Register Bit Functions (Continued)

82530 TIMING

The SCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to CLK. Since CLK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to CLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the rising edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the SCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the SCC. This time, T_{REC} must be at least 6 CLK cycles plus 130 ns, for the 82530-6.

Read Cycle Timing

Figure 11 illustrates Read cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTA}$ must remain stable throughout the cycle. If $\overline{CS}$ falls after $\overline{RD}$ falls or if it rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

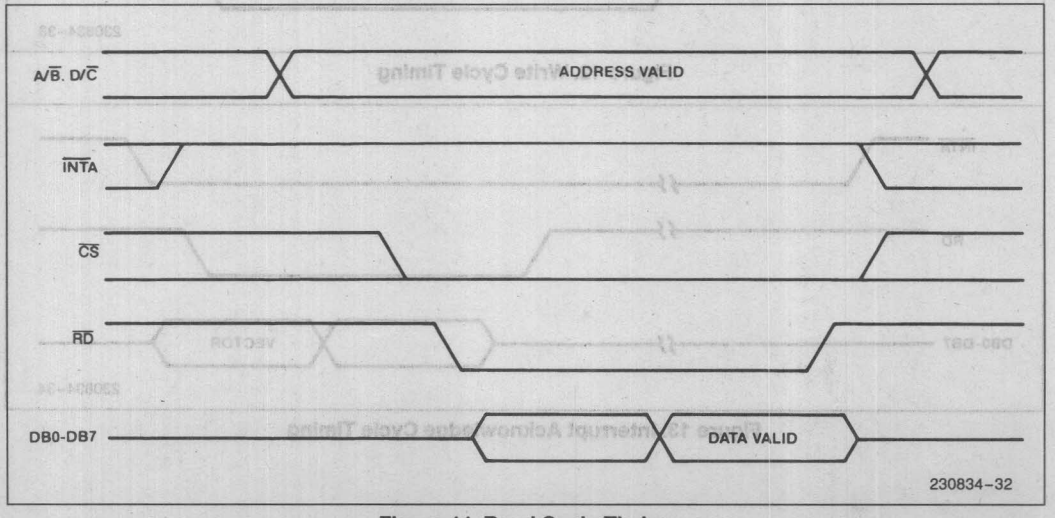


Figure 11. Read Cycle Timing

Write Cycle Timing

Figure 12 illustrates Write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTA}$ must remain stable throughout the cycle. If $\overline{CS}$ falls after $\overline{WR}$ falls or if it rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened.

Interrupt Acknowledge Cycle Timing

Figure 13 illustrates Interrupt Acknowledge cycle timing. Between the time $\overline{INTA}$ goes Low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{RD}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0–D_7 and it then sets the appropriate Interrupt-Under-Service internally.

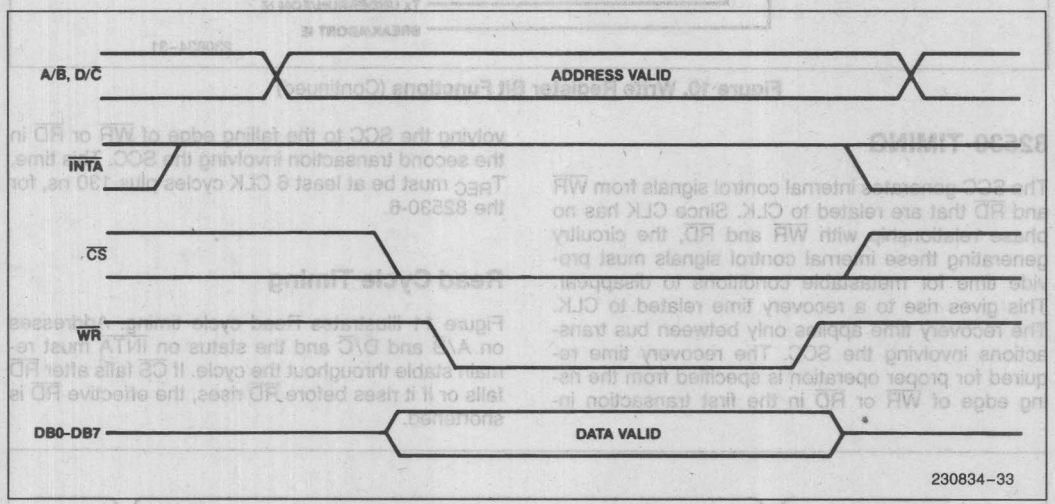


Figure 12. Write Cycle Timing

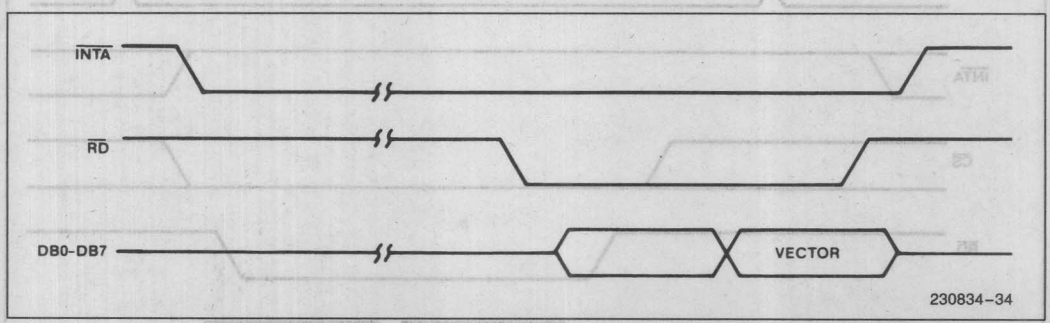


Figure 13. Interrupt Acknowledge Cycle Timing

ABSOLUTE MAXIMUM RATINGS*

Case Temperature
Under Bias . . . . . 0°C to +70°C

Storage Temperature
Ceramic Package . . . . . −65°C to +150°C
Plastic Package . . . . . −40°C to +125°C

Voltage on Any Pin with
Respect to Ground . . . . . −0.5V to +7.0V

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS T_C = 0°C to 70°C; V_{CC} = +5V ±5%

Symbol	Parameter	Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.4	V_{CC} +0.3	V	
V_{OL}	Output Low Voltage		+0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = −250 μA
I_{IL}	Input Leakage Current		±10	μA	0.4V to 2.4V
I_{OL}	Output Leakage Current		±10	μA	0.4V to 2.4V
I_{CC}	V_{CC} Supply Current		250	mA	

CAPACITANCE T_C = 25°C; V_{CC} = GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
C_{IN}	Input Capacitance		10	pF	f_c = 1 MHz
C_{OUT}	Output Capacitance		15	pF	Unmeasured pins returned to GND
$C_{I/O}$	Input/Output Capacitance		20	pF	

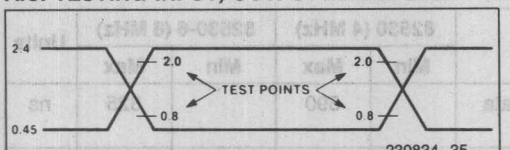
A.C CHARACTERISTICS $T_C = 0°C$ to $+70°C$; $V_{CC} = +5V \pm 5\%$

READ AND WRITE TIMING

Number	Symbol	Parameter	82530 (4 MHz)		82530-6 (6 MHz)		Units
			Min	Max	Min	Max	
1	tCL	CLK Low Time	105	2000	70	1000	ns
2	tCH	CLK High Time	105	2000	70	1000	ns
3	tf	CLK Fall Time		20		10	ns
4	tr	CLK Rise Time		20		15	ns
5	tCY	CLK Cycle Time	250	4000	165	2000	ns
6	tAW	Address to $\overline{WR}\downarrow$ Setup Time	80		0		ns
7	tWA	Address to $\overline{WR}\uparrow$ Hold Time	0		0		ns
8	tAR	Address to $\overline{RD}\downarrow$ Setup Time	80		0		ns
9	tRA	Address to $\overline{RD}\uparrow$ Hold Time	0		0		ns
10	tIC	$\overline{INTA}$ to CLK $\uparrow$ Setup Time	5		5		ns
11	tIW	$\overline{INTA}$ to $\overline{WR}\downarrow$ Setup Time (Note 1)	200		55		ns
12	tWI	$\overline{INTA}$ to $\overline{WR}\uparrow$ Hold Time	0		0		ns
13	tIR	$\overline{INTA}$ to $\overline{RD}\downarrow$ Setup Time (Note 1)	200		55		ns
14	tRI	$\overline{INTA}$ to $\overline{RD}\uparrow$ Hold Time	0		0		ns
15	tCI	$\overline{INTA}$ to CLK $\uparrow$ Hold Time	100		100		ns
16	tCLW	$\overline{CS}$ Low to $\overline{WR}\downarrow$ Setup Time	0		0		ns
17	tWCS	$\overline{CS}$ to $\overline{WR}\uparrow$ Hold Time	0		0		ns
18	tCHW	$\overline{CS}$ High to $\overline{WR}\downarrow$ Setup Time	100		5		ns
19	tCLR	$\overline{CS}$ Low to $\overline{RD}\downarrow$ Setup Time (Note 1)	0		0		ns
20	tRCS	$\overline{CS}$ to $\overline{RD}\uparrow$ Hold Time (Note 1)	0		0		ns
21	tCHR	$\overline{CS}$ High to $\overline{RD}\downarrow$ Setup Time (Note 1)	100		5		ns
22	tRR	$\overline{RD}$ Low Time (Note 1)	390		150		ns
23	Null	Parameter Deleted					
24	tRDI	$\overline{RD}\uparrow$ to Data Not Valid Delay	0		0		ns
25	tRDV	$\overline{RD}\downarrow$ to Data Valid Delay		250		105	ns
26	tDF	$\overline{RD}\uparrow$ to Output Float Delay (Note 2)		70		45	ns

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a +0.5V change in the output with a maximum D.C. load and minimum A.C. load.

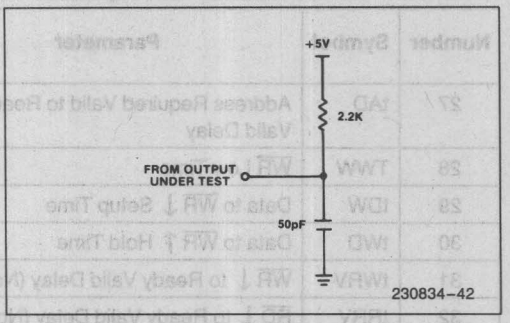
A.C. TESTING INPUT, OUTPUT WAVEFORM



230834-35

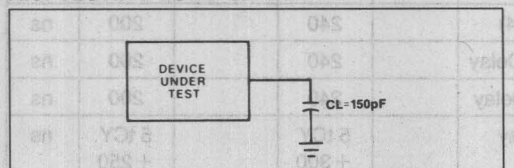
A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

OPEN DRAIN TEST LOAD



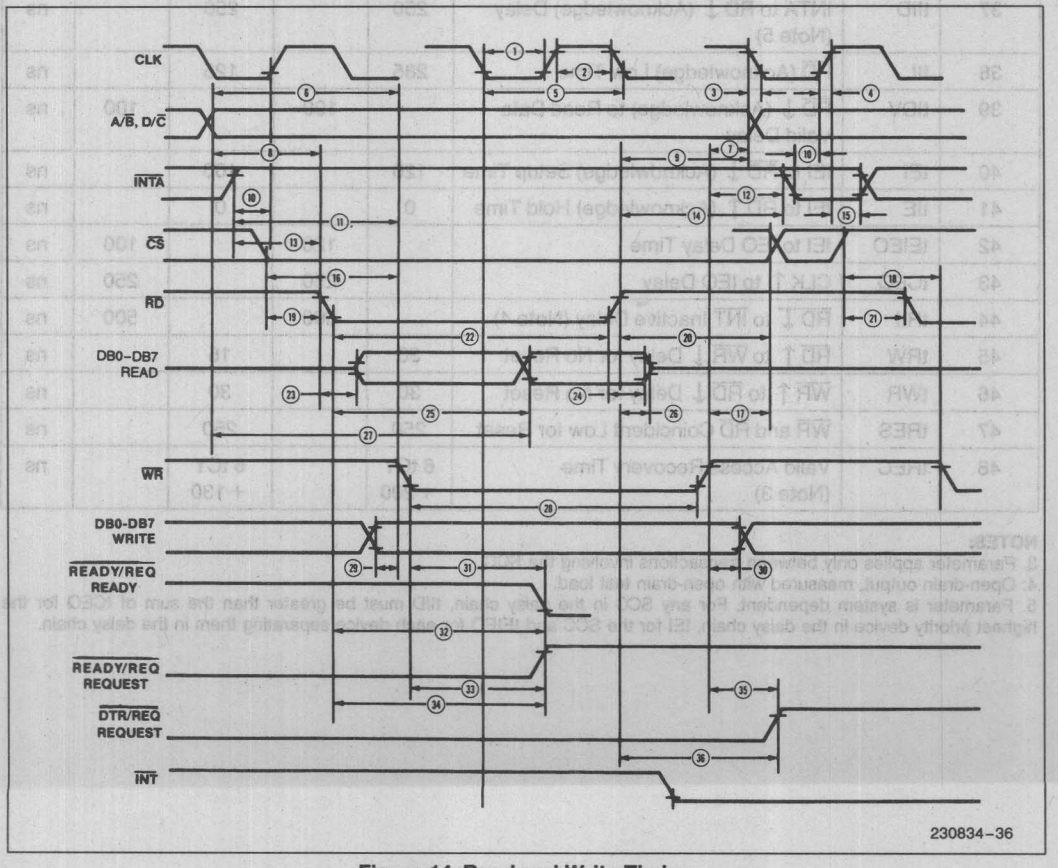
230834-42

A.C. TESTING LOAD CIRCUIT



230834-41

C_L = 150 pF
C_L Includes Jig Capacitance



230834-36

Figure 14. Read and Write Timing

INTERRUPT ACKNOWLEDGE TIMING, RESET TIMING, CYCLE TIMING

Number	Symbol	Parameter	82530 (4 MHz)		82530-6 (6 MHz)		Units
			Min	Max	Min	Max	
27	tAD	Address Required Valid to Read Data Valid Delay		590		325	ns
28	TWW	$\overline{WR}$ Low Time	390		60		ns
29	tDW	Data to $\overline{WR}\downarrow$ Setup Time	0		0		ns
30	tWD	Data to $\overline{WR}\uparrow$ Hold Time	0		0		ns
31	tWRV	$\overline{WR}\downarrow$ to Ready Valid Delay (Note 4)		240		200	ns
32	tRRV	$\overline{RD}\downarrow$ to Ready Valid Delay (Note 4)		240		200	ns
33	tWRI	$\overline{WR}\downarrow$ to $\overline{READY/REQ}$ Not Valid Delay		240		200	ns
34	tRRI	$\overline{RD}\downarrow$ to $\overline{READY/REQ}$ Not Valid Delay		240		200	ns
35	tDWR	$\overline{WR}\uparrow$ to $\overline{DTR/REQ}$ Not Valid Delay		5 tCY +300		5 tCY +250	ns
36	tDRD	$\overline{RD}\uparrow$ to $\overline{DTR/REQ}$ Not Valid Delay		5 tCY +300		5 tCY +250	ns
37	tIID	$\overline{INTA}$ to $\overline{RD}\downarrow$ (Acknowledge) Delay (Note 5)	250		250		ns
38	tII	$\overline{RD}$ (Acknowledge) Low Time	285		125		ns
39	tIDV	$\overline{RD}\downarrow$ (Acknowledge) to Read Data Valid Delay		190		100	ns
40	tEI	IEI to $\overline{RD}\downarrow$ (Acknowledge) Setup Time	120		100		ns
41	tIE	IEI to $\overline{RD}\uparrow$ (Acknowledge) Hold Time	0		0		ns
42	tEIEO	IEI to IEO Delay Time		120		100	ns
43	tCEQ	CLK $\uparrow$ to IEO Delay		250		250	ns
44	tRII	$\overline{RD}\downarrow$ to $\overline{INT}$ Inactive Delay (Note 4)		500		500	ns
45	tRW	$\overline{RD}\uparrow$ to $\overline{WR}\downarrow$ Delay for No Reset	30		15		ns
46	tWR	$\overline{WR}\uparrow$ to $\overline{RD}\downarrow$ Delay for No Reset	30		30		ns
47	tRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset	250		250		ns
48	tREC	Valid Access Recovery Time (Note 3)	6 tCY +200		6 tCY +130		ns

NOTES:
3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, tIID must be greater than the sum of tCEQ for the highest priority device in the daisy chain, tEI for the SCC and tEIEO for each device separating them in the daisy chain.

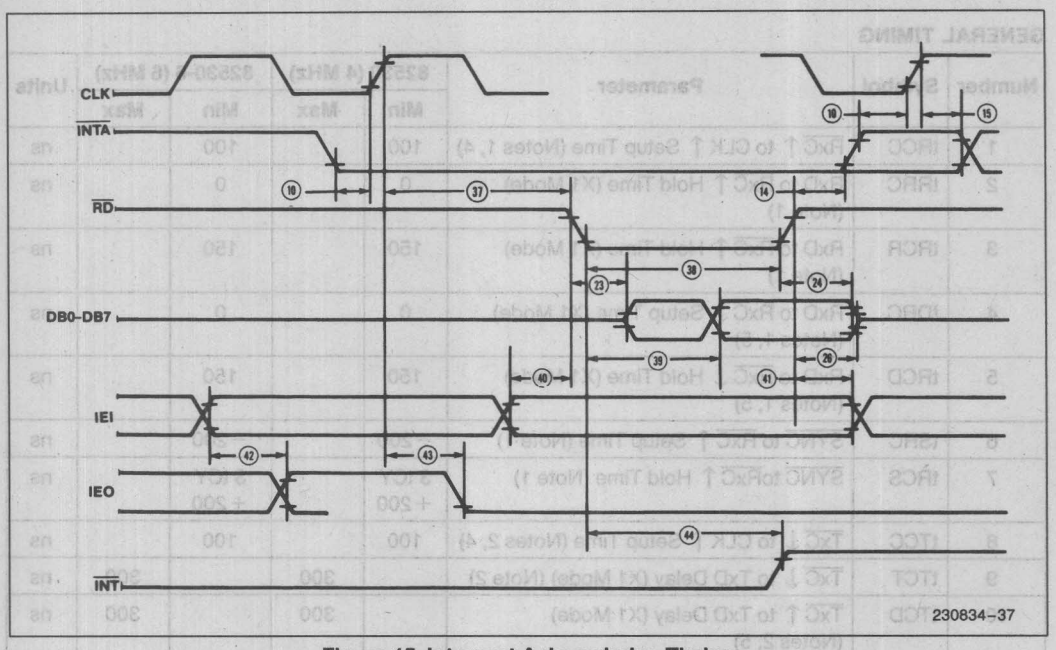


Figure 15. Interrupt Acknowledge Timing

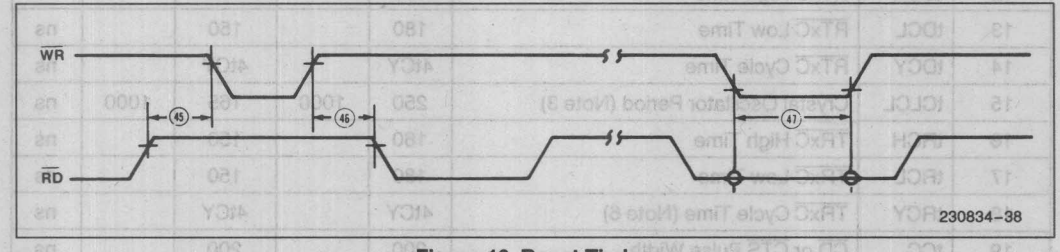


Figure 16. Reset Timing

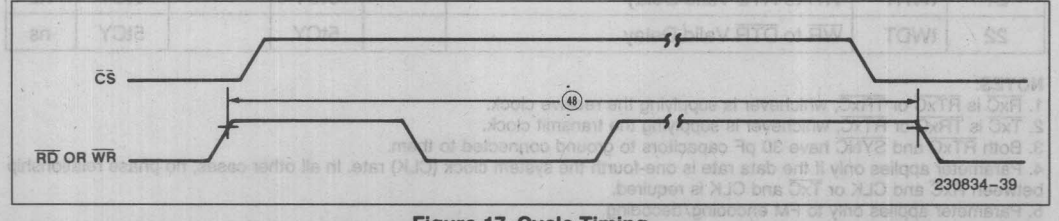


Figure 17. Cycle Timing

GENERAL TIMING

Number	Symbol	Parameter	82530 (4 MHz)		82530-6 (6 MHz)		Units
			Min	Max	Min	Max	
1	tRCC	$\overline{RxC}$ ↑ to CLK ↑ Setup Time (Notes 1, 4)	100		100		ns
2	tRRC	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode) (Note 1)	0		0		ns
3	tRCR	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode) (Note 1)	150		150		ns
4	tDRC	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode) (Notes 1, 5)	0		0		ns
5	tRCD	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode) (Notes 1, 5)	150		150		ns
6	tSRC	$\overline{SYNC}$ to $\overline{RxC}$ ↑ Setup Time (Note 1)	−200		−200		ns
7	tRCS	$\overline{SYNC}$ to$\overline{RxC}$ ↑ Hold Time (Note 1)	3 tCY +200		3 tCY +200		ns
8	tTCC	$\overline{TxC}$ ↓ to CLK ↑ Setup Time (Notes 2, 4)	100		100		ns
9	tTCT	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode) (Note 2)		300		300	ns
10	tTCD	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode) (Notes 2, 5)		300		300	ns
11	tTDT	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)		200		200	ns
12	tDCH	$\overline{RTxC}$ High Time	180		150		ns
13	tDCL	$\overline{RTxC}$ Low Time	180		150		ns
14	tDCY	$\overline{RTxC}$ Cycle Time	4tCY		4tCY		ns
15	tCLCL	Crystal Oscillator Period (Note 3)	250	1000	165	1000	ns
16	tRCH	$\overline{TRxC}$ High Time	180		150		ns
17	tRCL	$\overline{TRxC}$ Low Time	180		150		ns
18	tRCY	$\overline{TRxC}$ Cycle Time (Note 6)	4tCY		4tCY		ns
19	tCC	$\overline{CD}$ or $\overline{CTS}$ Pulse Width	200		200		ns
20	tSS	$\overline{SYNC}$ Pulse Width	200		200		ns
21	tWRT	$\overline{WR}$ to $\overline{RTS}$ Valid Delay		6tCY		6tCY	ns
22	tWDT	$\overline{WR}$ to $\overline{DTR}$ Valid Delay		5tCY		5tCY	ns

NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{SYNC}$ have 30 pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the system clock (CLK) rate. In all other cases, no phase relationship between $\overline{RxC}$ and CLK or $\overline{TxC}$ and CLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Only applies to transmitter and receiver. For DPLL and Baud Rate Generator Timings, the requirements are identical to system clock, CLK, specifications.

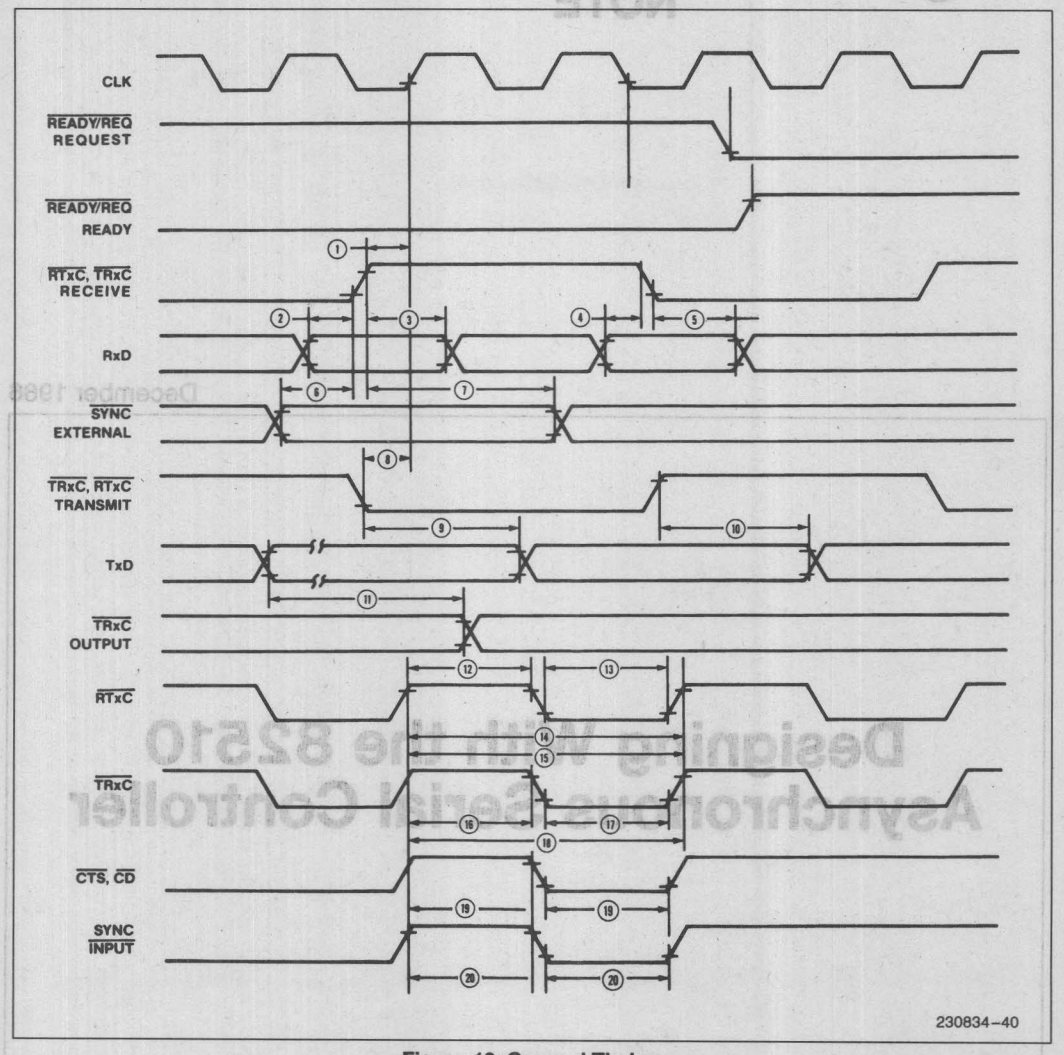
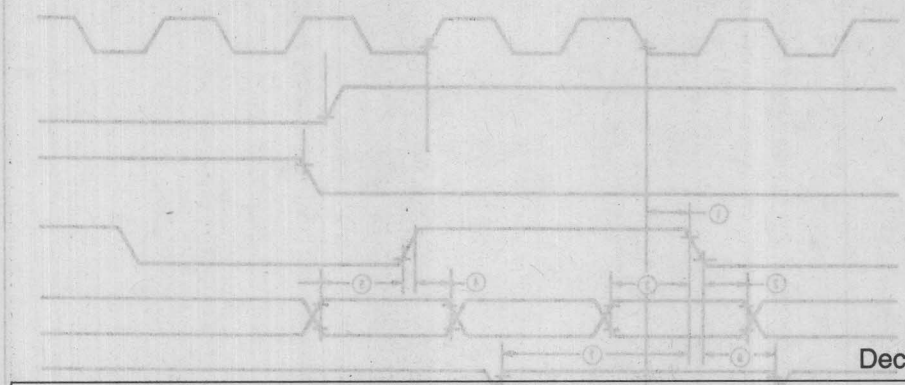


Figure 18. General Timing

intel®

APPLICATION NOTE

AP-401



December 1986

Designing With the 82510 Asynchronous Serial Controller

FAISAL IMDAD-HAQUE
APPLICATIONS ENGINEER

Order Number: 231928-001

DESIGNING WITH THE 82510 ASYNCHRONOUS SERIAL CONTROLLER

CONTENTS

PAGE

1.0 INTRODUCTION 2-184
1.1 Goal 2-184
1.2 Scope 2-184

2.0 82510 DESCRIPTION 2-184
2.1 Overview 2-184
2.2 Bus Interface Unit (BIU) 2-184
2.3 Receive Machine (RXM) 2-184
2.4 Transmit Machine (TXM) 2-185
2.5 Modem Interface Module 2-185
2.6 Timing Unit 2-185
2.7 FIFOs, Rx and Tx 2-186

3.0 HARDWARE DESIGN 2-186
3.1 System Interface 2-186
3.2 Reset 2-194
3.3 System Clock Options 2-195

4.0 INTERRUPT BEHAVIOR 2-197
4.1 FIFO Usage 2-197
4.2 Interrupt Handling 2-198
4.3 Polling 2-205

5.0 SOFTWARE CONSIDERATIONS 2-205
5.1 Configuration 2-205
5.2 Transmit Operation 2-208
5.3 Data Reception 2-213
5.4 Timer Usage 2-218

6.0 82510 IMPLEMENTATION OF XMODEM 2-221
6.1 XMODEM Protocol 2-221
6.2 Software 2-221
6.3 Software Listings 2-229

1.0 INTRODUCTION

The emergence of asynchronous communications as the most widely used protocol (it commands the largest installed base of nodes, exceeding HDLC/SDLC, the second most popular protocol, by a factor of 10 to 1) for point to point serial links has led to the need for an asynchronous communications component with high integration to reduce component count and decrease the cost of a serial port. The trend towards higher data rates and multiple job, multiple user systems has underscored the need for an intelligent serial controller to improve system throughput and decrease the CPU load normally associated with asynchronous serial communications.

The 82510 CMOS Asynchronous Serial Controller is designed to improve asynchronous communications throughput and reduce system cost by integrating functions and simplifying the system interface. Two independent FIFOs, and Control Character Recognition (CCR), provide data buffering and increase software efficiency. Two Baud Rate Generators/Timers, an On-Chip Crystal Oscillator and seven Programmable I/O pins provide a high degree of integration and reduce system component count. This application note will demonstrate the use of these features in an Asynchronous Communications Environment.

1.1 Goal

The goal of this application note is to demonstrate the use of the major 82510 features in an asynchronous communications environment and to depict basic hardware and software design techniques for the 82510. It will discuss interfaces using both polling and interrupt techniques, as well as the impact of FIFOs using either scheme. An application example covering the application of Error Free File Transfer is also provided.

1.2 Scope

The application note describes the operation of the 82510 ASC in a normal (non 8051 9-bit) asynchronous communications mode. The majority of the discussion is focused towards the systems aspects of the Controller. The use of the 82510 in a multidrop or 8051 9-bit asynchronous environment is not covered. This application note assumes that the reader is familiar with the 82510 in terms of pin description, register architecture and interrupt structure. It is also assumed that the reader is familiar with the information provided in the 82510 Data Sheet.

The initial sections of the application note provide an overview of the 82510 and its major functional blocks. This is followed by a discussion of the hardware design and system interface considerations in sections three and four. The fifth section provides some software techniques for transmitting and receiving data as well as the use of timers. Section seven briefly discusses the file transfer application based on the XMODEM protocol and includes the software listings.

2.0 82510 DESCRIPTION

2.1 Overview

The 82510 can be divided into seven functional blocks (See Figure 1): Bus Interface Unit (BIU), Timing Unit, Modem Interface Module, Tx FIFO, Rx FIFO, Tx Machine and Rx Machine. All blocks, except BIU can generate a block interrupt request to the 82510 interrupt logic. In the case of the Rx Machine, Timing Unit and Modem Interface Module, multiple sources (errors and status events) within the block cause the block interrupt request to become active. All of the blocks have registers associated with them. The registers, allow configuration, provide status information about events/errors, and may also be used to send commands to each block.

2.2 Bus Interface Unit (BIU)

The Bus Interface Unit (BIU) interfaces the 82510 functional blocks to the system or CPU bus. It provides read and write access to the 82510 registers and controls the generation of interrupts to the external world. The interrupt logic resolves contention between block interrupt requests, on a priority basis. The BIU also has the Hardware Reset circuitry, which is driven by the RESET pin. The reset signal clears all internal Flip Flops, and Registers and puts them in a predefined state. All activities on the Bus interface, including register accesses by the CPU, are synchronized to an internal (82510) system clock, supplied via the CLK pin.

2.3 Receive Machine (RxM)

The Rx Machine (RxM) converts the serial data to parallel and writes it to the Rx FIFO, along with the appropriate flags (available in the *Receive Flags Register*). The Rx Machine can be configured for control character recognition, data sampling and DPLL operation. The software can check for noise, control character, break, address or parity and framing errors by reading the status or character flags. Optionally, the Receive Status bits (in RST), when enabled, can generate interrupt requests. The Rx Machine block Interrupt request is reflected in the *General Status Register* and is set when an enabled interrupt request within the Rx Ma-

chine (i.e. RST bits) becomes active. The Rx Machine has eight registers associated with it:

Receive Data (RXD)—Receive Data Character

Receive Flags (RXF)—Receive Character Flags

Receive Status (RST)—Receive Events and Receive Errors

Receive Interrupt Enable (RIE)—Enables Interrupts on corresponding bits in RST

Receive Mode (RMD)—Receive Machine Configuration

Receive Command (RCM)—Receiver Command Register

Line Control (LCR)—16450 Register, Character Attribute Configuration

Line Status (LSR)—16450 Status Register, Tx and Rx status

2.4 Transmit Machine (TxM)

The Tx Machine reads characters from the Tx FIFO and transmits them serially over the TXD line. The Tx Machine can also transmit additional character attributes (9th bit of Data, Address Marker, Software Parity) available from the Transmit Flags, if configured in the appropriate mode. The Tx Machine Idle interrupt request is reflected in the GSR and LSR registers to indicate that the Transmitter is either Empty or Disabled. The Tx Machine has six registers associated with it:

Line Control (LCR)—16450 Register, Character Attribute Configuration

Line Status (LSR)—16450 Status Register, Tx and Rx status

Transmit Mode (TMD)—Tx Machine Configuration

Transmit Command—(TCM)—Transmit Command Register

Transmit Flags (TXF)—Transmit Character Flags

Transmit Data (TXD)—Transmit Data Character

2.5 Modem Interface Module

The Modem Interface module is responsible for the modem interface and general purpose I/O pins. It will generate Interrupts (if enabled) upon transitions in the modem input pins ($\overline{DCD}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DSR}$). The modem output pins can be controlled by the CPU, also the RTS pin can be used to provide flow control, in the automatic transmission mode. It is the source of the Modem Interrupt bit in GSR. This bit is set whenever there is a state change in the $\overline{DCD}$, $\overline{RI}$, $\overline{DSR}$ or $\overline{CTS}$ inputs (reflected in *Modem Status Register*) and the corresponding enable bits are set. The function and direction of the multifunction pins can be reprogrammed and is available as a configuration option. Multifunction pins, when configured as outputs, can be controlled by the CPU through the *Modem Control Register.* The Modem module has four registers associated with it:

Modem Status Register (MSR)—State transitions on modem input pins, and State of the modem input pins

Modem Control (MCR)—Control state of Modem Output pins

Modem Interrupt Enable (MIE)—Enable Interrupt on State transitions in modem input pins

I/O Pin Mode (PMD)—Functions and Directions of Multifunction pins

2.6 Timing Unit

The Timing Unit is responsible for the generation of the System Clock, using either its Crystal Oscillator or an externally generated clock, and generation of the Tx and Rx clocks from either the On-Chip Baud Rate Generators or the SCLK pin. It is also responsible for generating Timer Expired interrupts when the BRGs/Timers are configured for use as Timers. There are ten registers associated with the Timing Unit, four of these are used in the Timer mode only.

Timer Status (TMST)—Timer A and/or Timer B expired

Timer Interrupt Enable (TMIE)—Enables Interrupts upon Expiration of Timers A or B in TMST

Timer Control (TMCR)—Start and Disable Timers

Clock Configure (CLCF)—Select source and mode for Tx and Rx clocks

BRG B Configuration (BBCF)—Mode and Clock source of BRG B

BRG B LSB of Divisor (BBL)—Least Significant Byte of BRG B Divisor/Count

BRG A MSB of Divisor (BBH)—Most Significant Byte of BRG B Divisor/Count

BRG A Configuration (BACF)—Mode and Clock source of BRG A

BRG A LSB of Divisor (BAL)—Least Significant Byte of BRG A Divisor/Count

BRG A MSB of Divisor (BAH)—Most Significant Byte of BRG A Divisor/Count

2.7 FIFOs, Rx and Tx

The Dual FIFOs (transmit and receive), serve as buffers for the 82510. They buffer the transmitter and Receiver from the CPU. Each of the FIFOs has a programmable threshold. The threshold is the FIFO level which will generate an interrupt. The threshold is used to optimize the CPU throughput and provide increased interrupt to service latency for higher baud rates. It can be configured through the FIFO Mode Register. Each FIFO character has flags associated with it (TxF and RxF). As each character is read from the Rx FIFO its flags are put into the RxF register. Before a write to TXD (if character configuration requires) the character flags are written to the TXF register. The two FIFOs are totally independent of each other and each FIFO can generate an interrupt request which indicates that the configured threshold has been met.

3.0 HARDWARE DESIGN

3.1 System Interface

The 82510 has a standard I/O peripheral interface, it has a demultiplexed Bus, which consists of a bidirectional eight bit Data Bus, and three Address lines. Interrupt, Read, Write, Chip Select and Reset pins complete the system interface. The three address lines along with the *Bank register* are used to select a particular register.

3.1.1 REGISTER ACCESS

The 82510 registers are logically divided into four banks. Only one bank can be accessed at any one time. Each register bank occupies eight I/O addresses. To select a register, the correct Bank must first be selected by writing to the GIR/Bank register (the *GIR/Bank register* I/O address is two ($A_0 = 0, A_1 = 1, A_2 = 0$). Then one of the eight I/O space addresses is selected by outputting a value (between zero and seven) to the 82510 address pins A_0–A_2.

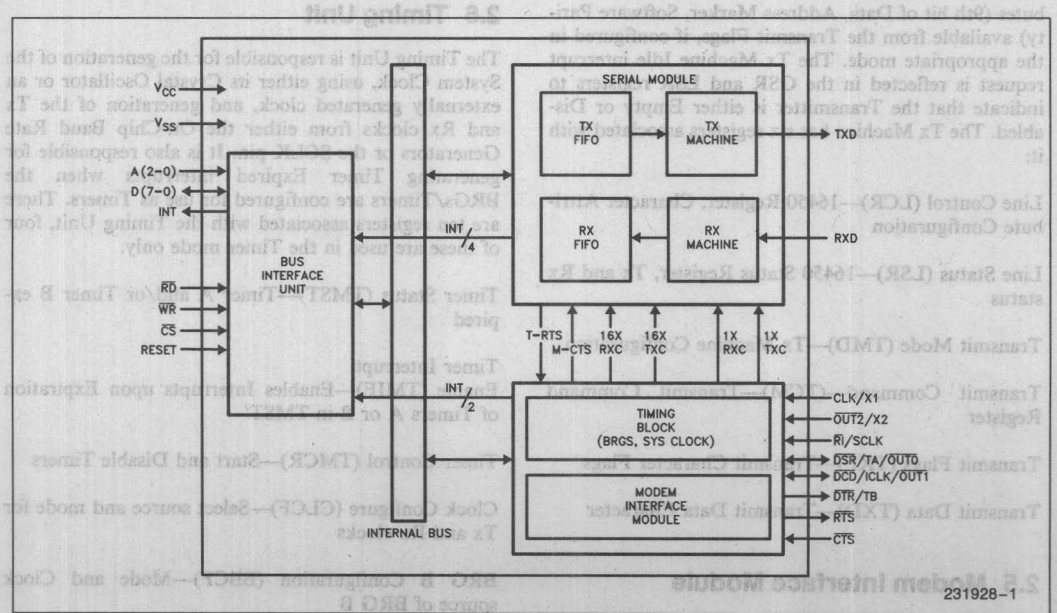


Figure 1. 82510 Block Diagram

BANK ZERO 8250—COMPATIBLE BANK										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
TxD	Tx Data bit 7	Tx Data bit 6	Tx Data bit 5	Tx Data bit 4	Tx Data bit 3	Tx Data bit 2	Tx Data bit 1	Tx Data bit 0	0	—
RxD	Rx Data bit 7	Rx Data bit 6	Rx Data bit 5	Rx Data bit 4	Rx Data bit 3	Rx Data bit 2	Rx Data bit 1	Rx Data bit 0	0	—
BAL	BRGA LSB Divide Count (DLAB = 1)								0	02H
BAH	BRGA MSB Divide Count (DLAB = 1)								1	00H
GER	0	0	Timer Interrupt Enable	Tx Machine Interrupt Enable	Modem Interrupt Enable	Rx Machine Interrupt Enable	Tx FIFO Interrupt Enable	Rx FIFO Interrupt Enable	1	00H
GIR/BANK	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
LCR	DLAB Divisor Latch Access bit	Set Break	Parity Mode bit 2	Parity Mode bit 1	Parity Mode bit 0	Stop bit Length bit 0	Character Length bit 1	Character Length bit 0	3	00H
MCR	0	0	OUT 0 Complement	Loopback Control bit	OUT 2 Complement	OUT 1 Complement	RTS Complement	DTR Complement	4	00H
LSR	0	TxM Idle	Tx FIFO Interrupt	Break Detected	Framing Error	Parity Error	Overrun Error	Rx FIFO Int Reg	5	60H
MSR	DCD Input Inverted	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State (H → L) Change in RI	State Change in DSR	State Change in CTS	6	00H
ACR0	Address or Control Character Zero								7	00H

BANK ONE—GENERAL WORK BANK										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
TxD	Tx Data bit 7	Tx Data bit 6	Tx Data bit 5	Tx Data bit 4	Tx Data bit 3	Tx Data bit 2	Tx Data bit 1	Tx Data bit 0	0	—
RxD	Rx Data bit 7	Rx Data bit 6	Rx Data bit 5	Rx Data bit 4	Rx Data bit 3	Rx Data bit 2	Rx Data bit 1	Rx Data bit 0	0	—
RxF	—	Rx Char OK	Rx Char Noisy	Rx Char Parity Error	Address or Control Character	Break Flag	Rx Char Framing Error	Ninth Data bit of Rx Char	1	—
TxF	Address Marker bit	Software Parity bit	Ninth bit of Data Char	0	0	0	0	0	1	—
GIR/BANK	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
TMST	—	—	Gate B State	Gate A State	—	—	Timer B Expired	Timer A Expired	3	30H
TMCR	0	0	Trigger Gate B	Trigger Gate A	0	0	Start Timer B	Start Timer A	3	—
MCR	0	0	OUT 0 Complement	Loopback Control bit	OUT 2 Complement	OUT 1 Complement	RTS Complement	DTR Complement	4	00H

Figure 2. 82510 Register Map

BANK ONE—GENERAL WORK BANK (Continued)										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
FLR	—		Rx FIFO Level		—		Tx FIFO Level		4	00H
RST	Address/ Control Character Received	Address/ Control Character Match	Break Terminated	Break Detected	Framing Error	Parity Error	Overrun Error	Rx FIFO Interrupt Requested	5	00H
RCM	Rx Enable	Rx Disable	Flush RxM	Flush Rx FIFO	Lock Rx FIFO	Open Rx FIFO	0	0	5	—
MSR	DCD Complement	RI Input Inverted	DSR Input Inverted	CTS Input Inverted	State Change in DCD	State Change in RI	State Change in DSR	State Change in CTS	6	00H
TCM	0	0	0	0	Flush Tx Machine	Flush Tx FIFO	Tx Enable	Tx Disable	6	—
GSR	—	—	Timer Interrupt	TxM Interrupt	Modem Interrupt	RxM Interrupt	Tx FIFO Interrupt	Rx FIFO Interrupt	7	12H
ICM	0	0	0	Software Reset	Manual Int Acknowledge Command	Status Clear	Power Down Mode	0	7	—

BANK TWO—GENERAL CONFIGURATION										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
FMD	0	0	Rx FIFO Threshold		0	0	Tx FIFO Threshold		1	00H
GIR/BANK	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
TMD	Error Echo Disable	Control Character Echo Disable	9-bit Character Length	Transmit Mode		Software Parity Mode	Stop Bit Length		3	00H
IMD	0	0	0	0	Interrupt Acknowledge Mode	Rx FIFO Depth	ulan Mode Select	Loopback or Echo Mode of Operation	4	0CH
ACR1	Address or Control Character 1								5	00H
RIE	Address/ Control Character Recognition Interrupt Enable	Address/ Control Character Match Interrupt Enable	Break Terminate Interrupt Enable	Break Detect Interrupt Enable	Framing Error Interrupt Enable	Parity Error Interrupt Enable	Overrun Error Interrupt Enable	0	6	1EH
RMD	Address/Control Character Mode		Disable DPLL	Sampling Window Mode	Start bit Sampling Mode	0	0	0	7	00H

BANK THREE—MODEM CONFIGURATION										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
CLCF	Rx Clock Mode	Rx Clock Source	Tx Clock Mode	Tx Clock Source	0	0	0	0	0	00H
BACF	0	BRGA Clock Source	0	0	0	BRGA Mode	0	0	1	04H
BBL	BRGB LSB Divide Count (DLAB = 1)								0	05H
BBH	BRGB MSB Divide Count (DLAB = 1)								1	00H

Figure 2. 82510 Register Map (Continued)

BANK THREE—MODEM CONFIGURATION (Continued)										
Register	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Address**	**Default**
GIR/BANK	0	BANK Pointer bit 1	BANK Pointer bit 0	0	Active Block Int bit 2	Active Block Int bit 1	Active Block Int bit 0	Interrupt Pending	2	01H
BBCF	BRGB Clock Source		0	0	0	BRGB Mode	0	0	3	84
PMD	DCD/ICLK/ OUT 1 Direction	DCD/ICLK/ OUT 1 Function	DSR/TA/ OUT 0 Direction	DSR/TA/ OUT 0 Function	RI/SCLK Function	DTR/TB Function	0	0	4	FCH
MIE	0	0	0	0	DCD State Change Int Enable	RI State Change Int Enable	DSR State Change Int Enable	CTS State Change Int Enable	5	0FH
TMIE	0	0	0	0	0	0	Timer B Interrupt Enable	Timer A Interrupt Enable	6	00H

Figure 2. 82510 Register Map (Continued)

2

3.1.2 READ AND WRITE CYCLES

Like most other I/O based peripherals the Read and Write pins are used to access data in the 82510. Each read or write cycle has specified setup and hold times in order for the data to be transferred correctly to/from the 82510. The critical timings for the read cycle are:

1. Address Valid to Read Active (Tavrl)
2. Command Access Time to Data Valid (Trldv)
3. Command Active Width (Trlrh)

The less critical parameters are:

4. Address Hold to Read Inactive (Trhax)
5. Data Out Float Delay after Read Inactive (Trhdz)

The critical timings for the write cycle are:

1. Address Valid to Write Low (Tavwl)
2. Write Active Time (Twlwh)
3. Data Valid to Write Inactive (Tdvwh)

The less critical parameters are:

4. Address and Chip Select Hold Time After Write Inactive (Twhax)
5. Data Hold Time After Write Inactive (Twhdx)

These timings determine the number of wait states required for the 82510 and the CPU interface. The interfaces for some popular microprocessors are discussed in the following sections.

3.1.3 80186 INTERFACE

The exact interface is shown in Figure 3. The schematic shows the 80186 interface to the 82510 on a local bus. Although the Data Bus is buffered, it is possible to directly connect the 82510 to the 80186 data bus; because the Data Float Delay after read inactive is 40 ns for the 82510, which is well under the 85 ns requirement of the 80186. The timing equations for the interface are given below.

Read Cycle:

$$\text{Address to Read Low} = \text{Tclcl} - \text{Tclav}_{max} + \text{Tclrl}_{min} - \text{Latch Delay}_{max}$$

$$\text{Read Access Time} = 2\text{Tclcl} - \text{Tclrl}_{max} - \text{Tdvcl} - \text{Transceiver Delay}_{max}$$

$$\text{Read Active Time} = \text{Trlrh} = 2\text{Tclcl} - 46$$

Write Cycle:

$$\text{Address Valid to Write Active} = \text{Tclcl} + \text{Tcvctv}_{min} - \text{Tclav}_{max} - \text{Latch Prop. Delay}_{max}$$

$$\text{Write Active Time} = \text{Twlwh} = 2\text{Tclcl} - 40$$

$$\text{Data Valid to Write Inactive} = 2\,\text{Tclcl} - \text{Tcldv}_{max} - \text{Transceiver Delay}_{max} + \text{Tcvctx}_{min}$$

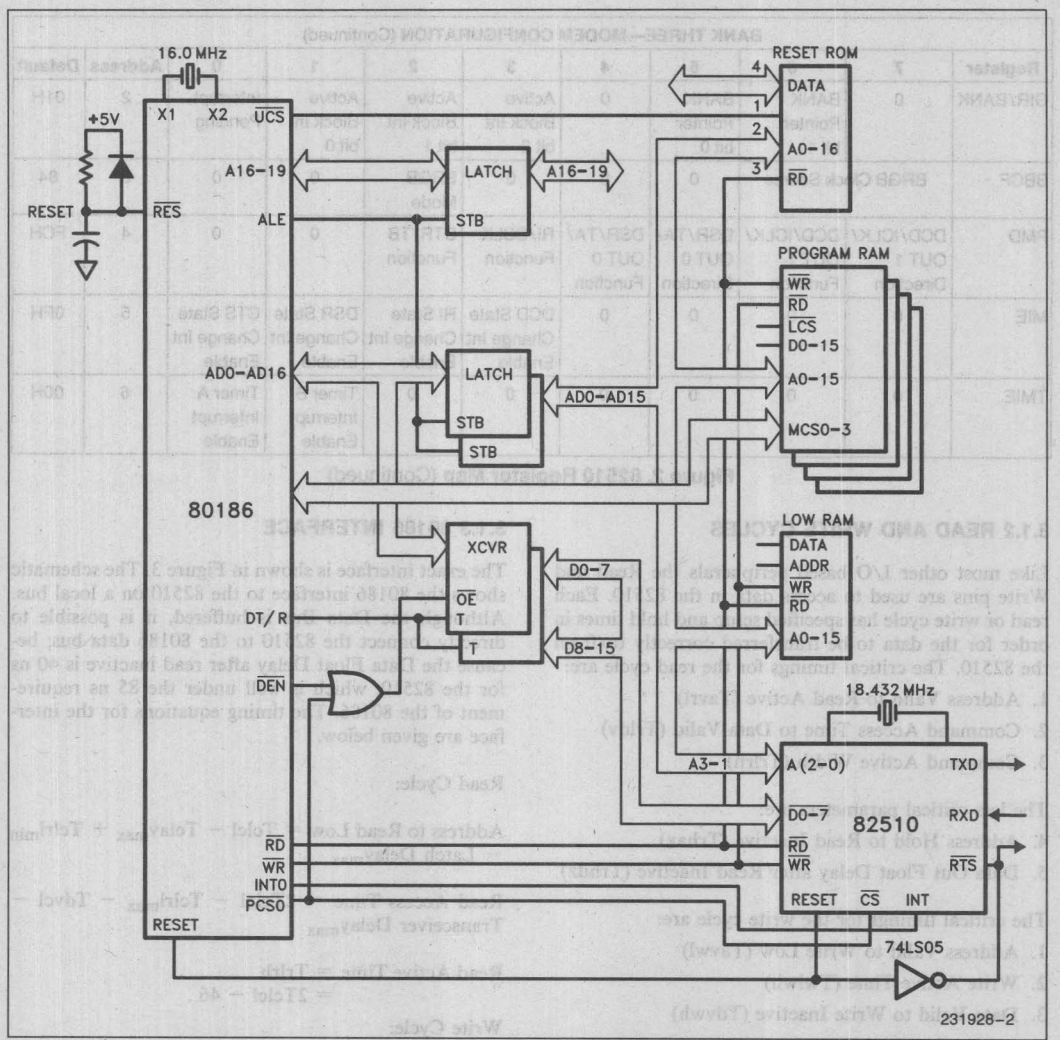


Figure 3. 82510 Interface to 80186

The user can transfer data to the 82510, using the DMA capabilities of the 80186, by using the $\overline{RTS}$ pin, in automatic modem control mode, as a DMA request line. The $\overline{RTS}$ pin, in automatic mode, will go inactive as soon as the Tx FIFO and the Tx shift register are empty. It will become active once a data character is written to the *TXD register.* In most 80186 DMA transfers the user has to make sure that the DMA request line goes inactive at least two clock cycles from the end of the DMA deposit cycle. In this case, the extra DMA cycle is not a problem, because the Tx FIFO will buffer the data to prevent an overrun (Since the Tx FIFO can buffer up to four characters, the $\overline{RTS}$ pin only needs to go inactive two clocks before the end of the deposit phase of the fourth DMA). Typically $\overline{RTS}$ will go inactive five (82510) system clocks after the rising edge of write.

3.1.4 80286 INTERFACE

The 80286 interface is shown in Figure 4. The 82510 is on the local bus, and is using the control signals from the 82288 Bus Controller. The Data Enable (OE) is qualified by the 82510 Chip Select, to avoid Data Bus contention between the 82510 and the CPU. The timing equations for the Read and Write Cycles are given below.

Read Cycle:

Address Valid to Read Active = T1 (CLK period) + $T29_{min}$ (CLK to cmd active) − $T16_{max}$ (ALE active delay) − Latch Prop. $Delay_{max}$

Read Access to Data Valid = 2T1 (CLK period) − $T29_{max}$ (CLK to cmd active) − T8 (Read Data Setup Time) − Transceiver $Delay_{max}$

Read Active Time = 2T1 (CLK period) − $T29_{max}$ (CLK to cmd active) + $T30_{min}$ (CLK to cmd inactive)

Write Cycle:

Address to Write Low = T1 (CLK period) + $T29_{min}$ (CLK to cmd active) − $T16_{max}$ (ALE active delay) − Latch $Delay_{max}$

Write Active Time = 2 T1 (CLK period) − $T29_{max}$ (CLK to cmd active) + $T30_{min}$ (CLK to cmd inactive)

Data to Write High = 3 T1 − $T14_{min}$ (Write Data Valid Delay) + $T30_{min}$ (CLK to cmd inactive) − Xcvr. $Delay_{max}$

Using an 8 MHz 80286 with the 82510 at 18.432 MHz (divide by two—9.216 MHz) requires two wait states. The critical timings are the read cycle timings—Read Access Time and Read Active Width. Inserting two wait states means that the access times for the relevant parameters will be increased by 250 ns.

NOTE:
The address decoding scheme of the 80286 interface is different from the IBM PC/PC AT I/O addresses for the serial ports, therefore the interface shown in Figure 4 cannot be used in PC/PC AT oriented designs.

3.1.5 80386 INTERFACE

The 80386 interface to the 82510 is given in Figure 5. The example uses the Basic I/O interface given in the 80386 Hardware Reference Manual section 8.3. The only differences are in the specific address lines used for chip select generation, and the additional wait states in the wait state generation logic. The address lines A3, A4 and A5 are used to select one of the eight register address spaces in the 82510, therefore, A6 and A7, rather than A4 and A5, are used in the I/O decoder. This causes a granularity of four in the 82510's I/O address space, i.e., the addresses of two consecutive registers in the 82510 differ by four.

The 82510 requires one additional wait state (as currently specified), the design assumes that the PAL equations are modified for that purpose. The user may also externally generate the wait states and connect to the "other ready logic" input ORed with the RDY pin of PAL 2. The two read timings Read Active width and Read Access time to Data Valid each require one additional wait state in order to meet the 82510 timing requirements. The timings are given below. (82510 times are at 9.216 MHz)

Read Cycle:

Read Access to Data Valid	= 253.25 ns
82510 Trldv	= 308
additional time reqd.	= 308-253.25
	= 54.75 ns
Read Active Width	= 269.25
82510 Trlrh	= 308
additional time reqd.	= 308-269.25
	= 38.75 ns
Address Valid to Read Active	= 132.75 ns
82510 T_{AVRL}	= 7 ns

Since each additional wait state adds 62.5 ns at 16 MHz, the 82510 requires one additional wait state.

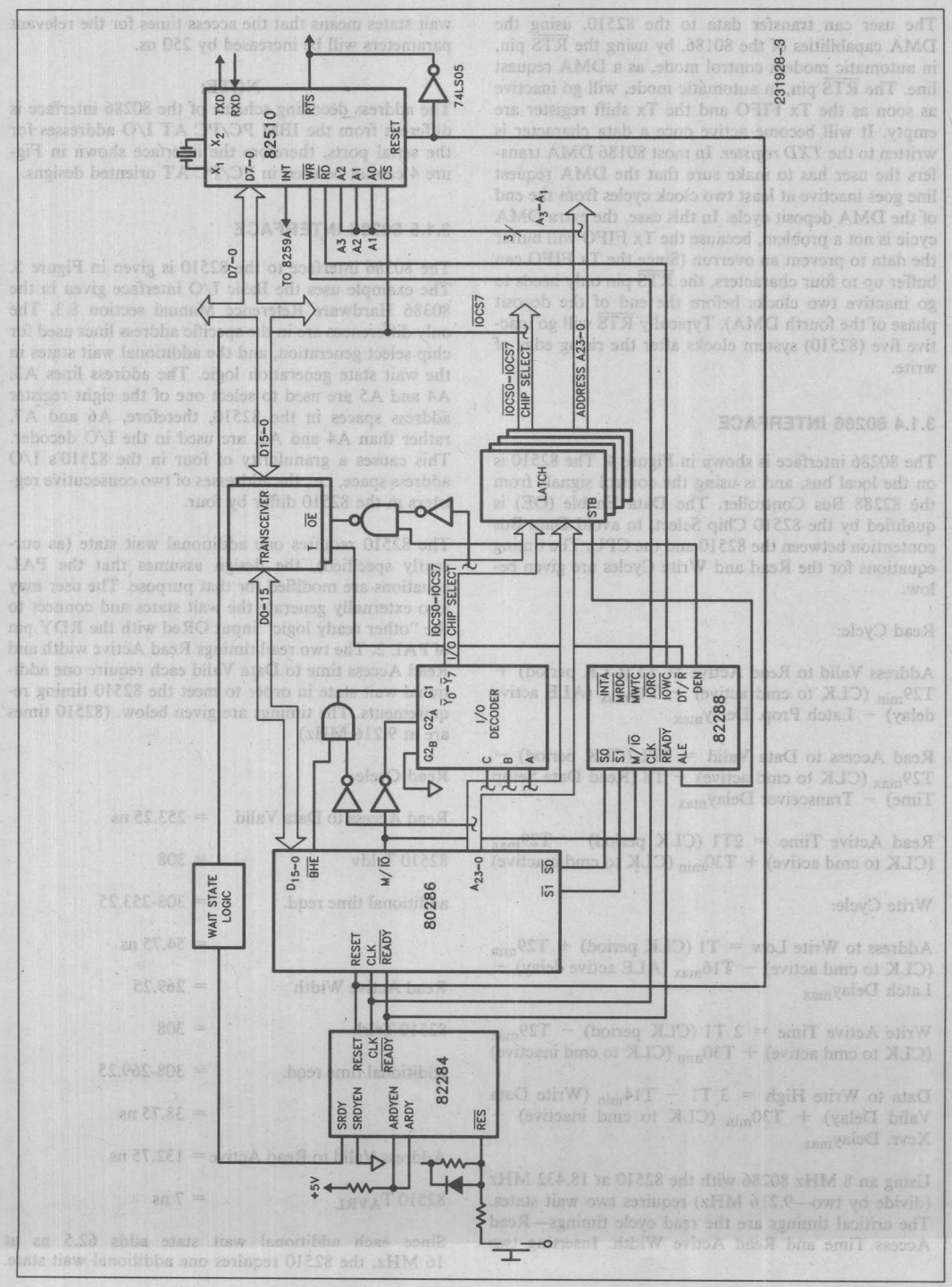


Figure 4. 80286 Interface to the 82510

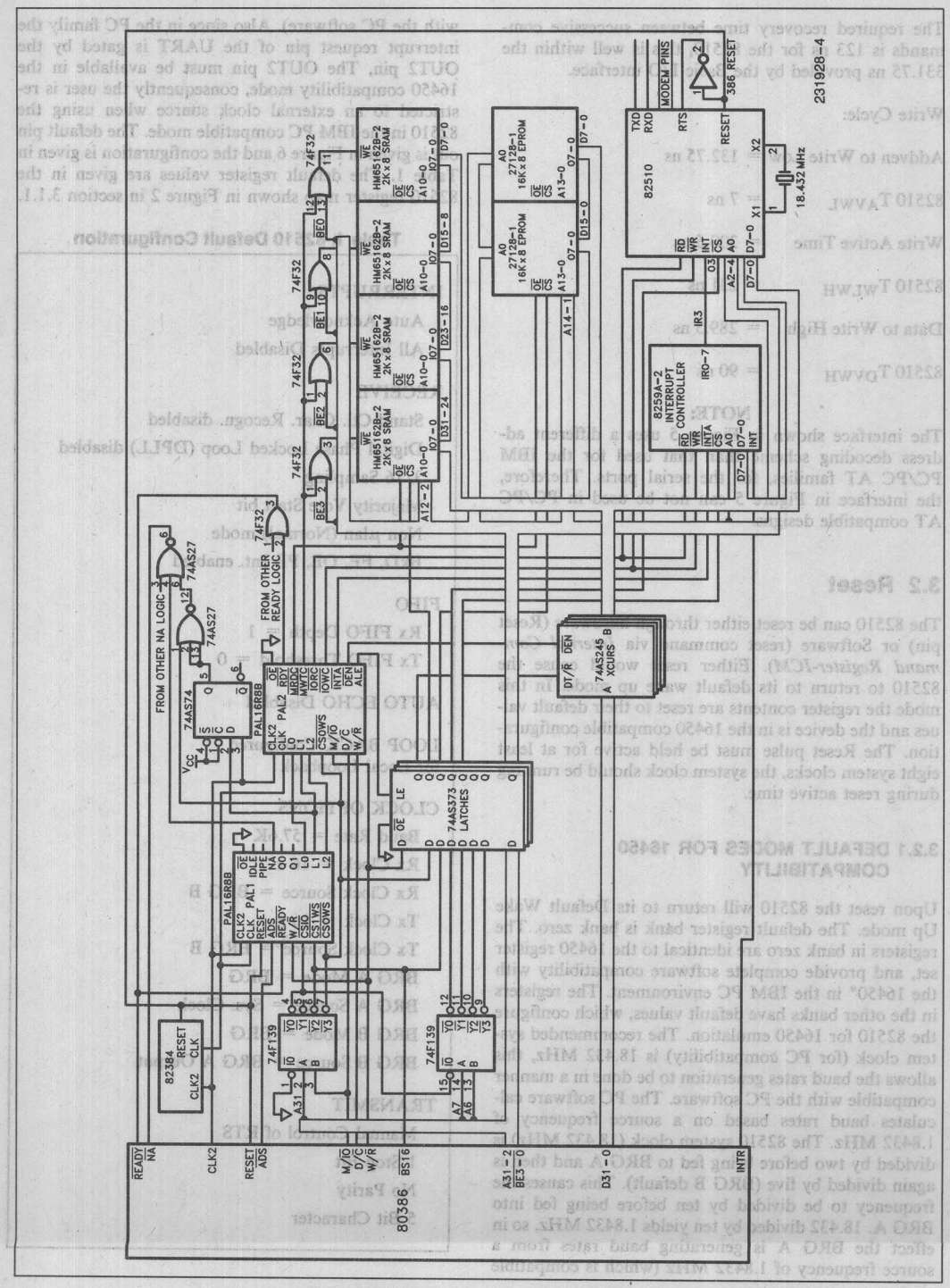


Figure 5. 80386 Interface to the 82510

NOTE:
The interface shown in Figure 5 uses a different address decoding scheme than that used for the IBM PC/PC AT families, for the serial ports. Therefore, the interface in Figure 5 can not be used in PC/PC AT compatible designs.

3.2 Reset

The 82510 can be reset either through hardware (Reset pin) or Software (reset command via *Internal Command Register-ICM*). Either reset would cause the 82510 to return to its default wake up mode. In this mode the register contents are reset to their default values and the device is in the 16450 compatible configuration. The Reset pulse must be held active for at least eight system clocks, the system clock should be running during reset active time.

3.2.1 DEFAULT MODES FOR 16450 COMPATIBILITY

Upon reset the 82510 will return to its Default Wake Up mode. The default register bank is bank zero. The registers in bank zero are identical to the 16450 register set, and provide complete software compatibility with the 16450* in the IBM PC environment. The registers in the other banks have default values, which configure the 82510 for 16450 emulation. The recommended system clock (for PC compatibility) is 18.432 MHz, this allows the baud rates generation to be done in a manner compatible with the PC software. The PC software calculates baud rates based on a source frequency of 1.8432 MHz. The 82510 system clock (18.432 MHz) is divided by two before being fed to BRG A and then is again divided by five (BRG B default). This causes the frequency to be divided by ten before being fed into BRG A. 18.432 divided by ten yields 1.8432 MHz, so in effect the BRG A is generating baud rates from a source frequency of 1.8432 MHz (which is compatible

RECEIVE
- Stand Ctl. Char. Recogn. disabled
- Digital Phase Locked Loop (DPLL) disabled
- 3/16 Sampling
- Majority Vote Start bit
- Non μlan (Normal) mode
- BkD, FE, OE, PE Int. enabled

FIFO
- Rx FIFO Depth = 1
- Tx FIFO Threshold = 0

AUTO ECHO Disabled

LOOP BACK Configured
for Local Loopback

CLOCK OPTIONS
- Baud Rate = 57.6K
- Rx Clock = 16 x
- Rx Clock Source = BRG B
- Tx Clock = 16 x
- Tx Clock Source = BRG B
- BRG A Mode = BRG
- BRG A Source = Sys. Clock
- BRG B Mode = BRG
- BRG B Source = BRG A Output

TRANSMIT
- Manual Control of RTS
- 1 Stop Bit
- No Parity
- 5 Bit Character

*16450 is the PC AT version of the INS 8250A.

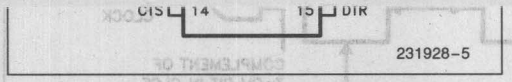


Figure 6. Default Pin Out Configuration of the 82510

3.3 System Clock Options

The term "System Clock" refers to the clock which provides timings for most of the 82510 circuitry. The 82510 has two modes of system clock usage. It can generate its system clock from its On-Chip Crystal Oscillator and an external crystal, or it can use an externally generated clock, input to the device through the CLK pin. The selection of the system clock option is done during reset. The default system clock source is an externally generated clock, which can be reconfigured by a strapping option on the $\overline{RTS}$ pin. During Reset, the $\overline{RTS}$ pin is an input; it is internally pulled high, if it is externally driven low, then the 82510 expects to use the Crystal Oscillator for system clock generation, otherwise it is set up for using an external clock source. This can be done by using an open collector inverter to $\overline{RTS}$, the input of the inverter is the Reset signal. The 82510 has a pull up resistor in the $\overline{RTS}$ circuitry so no external pull up is needed. In the crystal oscillator mode the CLK/X1 pin is automatically configured to X1, and the $\overline{OUT2}$/X2 pin is configured to X2. In the External Clock mode, the CLK/X1 is configured to CLK and the $\overline{OUT2}$/X2 is configured to $\overline{OUT2}$.

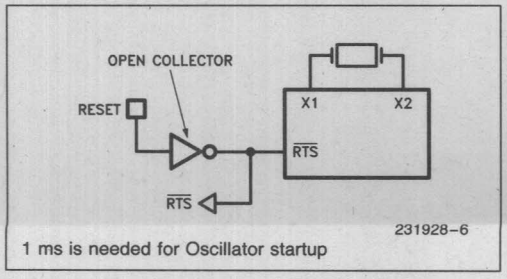


Figure 7. Crystal Oscillator Strapping Option

If the Crystal Oscillator is being used to supply the system clock, then the clock frequency is always divided by two before being fed into the rest of the 82510 circuitry. If, however an external clock source is being used to supply the system clock, then the user has two options:

1. Use the System Clock after **division by two,** e.g. if a 8 MHz clock is being fed into the CLK pin, then the actual frequency of the 82510 system clock will be 4 MHz (default).
2. **Disable Division by two** and use the direct undivided clock, e.g. if an 8 MHz clock is being fed into the CLK pin, then the actual frequency of the 82510 system clock is also 8 MHz.

The divide by two option is the default mode of operation in the External Clock mode of the 82510. A strapping option can be used to disable the Divide By Two operation (For Crystal Oscillator Mode Divide By Two must always be active). During Reset, the $\overline{DTR}$ pin is an input; it is internally pulled high, if it is externally driven low then the Divide By Two operation is disabled. The strapping option is identical to the one used on $\overline{RTS}$ for selection of the System Clock source.

The 82510 system clock must be chosen with care since it influences the wait state performance, Baud Rate Generation (if being used as source frequency for the BRGs), the power consumption, and the Timer counting period. The power consumption of the 82510 is dependent upon the system clock frequency. If using the system clock as a source for the Baud Rate Generator(s), then the system clock frequency must be a baud rate multiple in order to minimize frequency deviation. For standard baud rates a multiple of 1.8432 MHz can be used, in fact the 18.432 MHz maximum frequency was chosen with this particular criteria in mind.

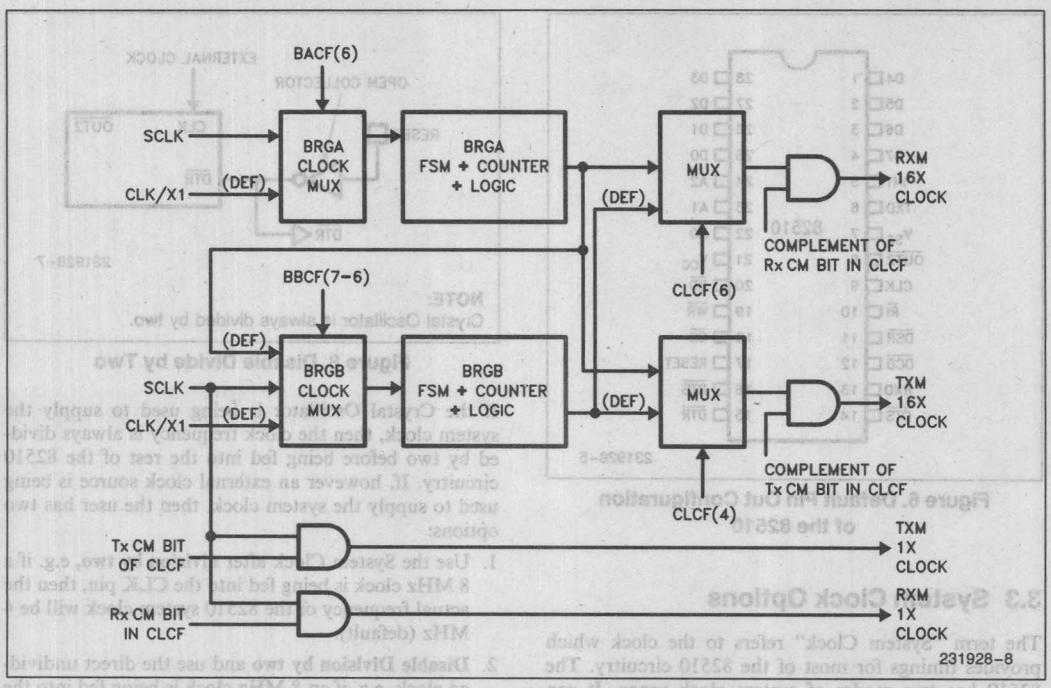


Figure 9. Timing Flow of the 82510

3.3.1 POWER DOWN MODE

The 82510 has a "power down" mode to reduce power consumption when the device is not in use. The 82510 powers down when the power down command is issued via the *Internal Command Register* (ICM). There are two modes of power down, Power Down Sleep and Power Down Idle.

3.3.1.1 Sleep Mode

This is the mode when even the system clock of the 82510 is shut down. The system clock source of the 82510 can either be the Crystal Oscillator or an external clock source. If the Crystal Oscillator is being used and the power down command is issued, then the 82510 will automatically enter the Sleep mode. If an external clock is being used, then the user must disable the external clock in addition to issuing the Power Down command, to enter the Sleep mode. The benefit of this mode is the increased savings in power consumption (typical power consumption in the Sleep mode is in the range of hundreds of microAmps. However, upon wake up, if using a crystal oscillator, the user must reprogram the device. The data is preserved if the external clock is disabled after the power down command, and enabled prior to exiting the power down mode. To exit this mode the user can either issue a Hardware reset, or read the *FIFO Level Register (FLR)* and then issue a software reset (if using a Crystal Oscillator). In either case the contents of the 82510 registers are not preserved and the device must be reprogrammed prior to operation.

NOTE:
If the Crystal Oscillator is being used then the user must allow about 1 ms for the oscillator to wake up before issuing the software reset.

3.3.1.2 Idle Mode

The 82510 is said to be in the Idle mode when the Power Down command is issued and the system clock is still running (i.e. the system clock is generated externally and not disabled by the user). In this mode the contents of all registers and memory cells are preserved, however, the power consumption in this mode is greater than in the Sleep mode. Reading FLR will take the 82510 out of this mode.

NOTE:
The data read from FLR when exiting Power Down is incorrect and must be ignored.

4.0 INTERRUPT BEHAVIOR

2

4.1 FIFO Usage

The 82510 has two independent four bytes transmit and receive FIFOs. Each FIFO can generate an interrupt request, when the FIFO level meets the Threshold requirements. The FIFOs can have a considerable impact on the performance of an asynchronous communications system. For systems using high baud rates they can provide increased interrupt-to-service latency reducing the chances of an overrun occurring. In systems constrained for CPU time, the FIFOs can increase the CPU Bandwidth by reducing the number of interrupt requests generated during asynchronous communications. It can reduce the interrupt load on the CPU by up to 75%. By choosing the FIFO thresholds which reflect the system bandwidth or service latency requirements, the user can achieve data rates and system throughput, unattainable with traditional UARTs.

Table 2. The Power Down Modes

Mode	Clock Source	Exit Procedure	Power Consumption	Data Preservation
Sleep	Crystal Oscill. Automatically Disabled	H/W Reset or Read FLR and Issue S/W Reset	100–900 μA	Not Preserved Must be Reprogrammed
	External Clock Must be Disabled by User	Enable External Clock, Read FLR and Issue S/W Reset H/W Reset	100–900 μA	Not Preserved Must be Reprogrammed
Idle	External Clock Running	H/W Reset Read FLR	1–3 mA	All Data Preserved Does Not Need to be Reprogrammed

4.1.1 INTERRUPT-TO-SERVICE LATENCY

The interrupt-to-service latency is the time delay from the generation of an interrupt request, to when the interrupt source in the 82510 is actually serviced. Its primary application is in the reception of data. In traditional UARTs the CPU must read the current character in the Receive Buffer before it is overrun by the next incoming character. The Rx FIFO in the 82510 can buffer up to four characters, allowing an interrupt-to-service latency of up to four character transmission times. The character transmission time is the time period required to transmit one full character at the given Baud Rate. It is dependent upon the baud rate and is given by equation (1):

(1) Character Transmission Time =

$$\frac{\text{Num. of Bits per Character Frame}}{\text{Baud Rate}}$$

The Transmit and Receive FIFO thresholds should be selected with consideration to two factors the Baud rate, and the (CPU Bandwidth allocated for Asynchronous Channels is dependent upon the number of channels supported since it does not include the overhead of supporting other peripherals) number of Asynchronous Serial ports being supported by the CPU. In order to avoid overrun, the interrupt-to-service delay must be less than the time it takes to fill the 82510 Rx FIFO. The relationship is given by equation (2):

(2) Int__to__service-latency < FIFO Size ×

Character Transmission Time

Example

Calculate the maximum baud rate that can be supported by a 6 MHz PC AT to support four Full Duplex Asynchronous channels using

a) The 82510 with four byte FIFO.

b) The 82510 with one byte FIFO.

Assumptions:

- CPU dedicated to Asynchronous communications.
- UART Interrupts limited to Transmission and Reception only.
- Interrupt Routines are optimized for fast throughput.
- 10 bits per character frame.

Going back to equation (2):

Int.__to__service latency < Buffer size x 10/baud rate

Int__to__service latency = # of Channels × (# of int. sources per channel) × Time required to service interrupt

Int__to__service latency = 4 × 2 × Time required to service interrupt

The Time required to service interrupt has been calculated to be 100 μs for a slightly optimized service routine. RMX86 interrupt service time is given as 250 μs and for other operating systems it should be slightly higher.

Int__to__service latency = 4x2x100 s
= 800 μs

82510 max Baud Rate = 4 × 10/800 μs
(four byte FIFO) = 50K bits/sec

82510 max Baud Rate = 1 × 10/800 μs
(one byte FIFO) = 12.5K bits/sec

4.2 Interrupt Handling

The 82510 has 16 different sources of interrupt, each of these sources, when set and enabled, will cause their respective block interrupt requests to go active. The block interrupt request, if enabled, will set the 82510's INT pin high, and will be reflected as a pending interrupt in the *General Interrupt Register (GIR)* if no other higher priority block is requesting service. If a higher priority block interrupt is also active at the same time, then the *General Interrupt Register* will reflect the higher priority request as the source of the 82510 interrupt. The lower priority interrupt will issue a new edge on the interrupt pin only after the higher priority interrupt is acknowledged and if no other priority block requests are present. Both the block interrupts and the individual sources within the blocks are maskable. The block interrupts are enabled through the *General Enable Register (GER)* which prevents masked bits in the *General Status Register (GSR)* from being decoded into the *General Interrupt Register.* This does not prevent the block request from being set in the *General Status Register*, it only prevents the masked GSR bits from being decoded into the *General Interrupt Register*, and thus generating any interrupts. The individual sources within the block are masked out via the corresponding interrupt enable register associated with the specific block (Rx Machine, Timing Unit and the Modem I/O module each have an Interrupt Enable register).

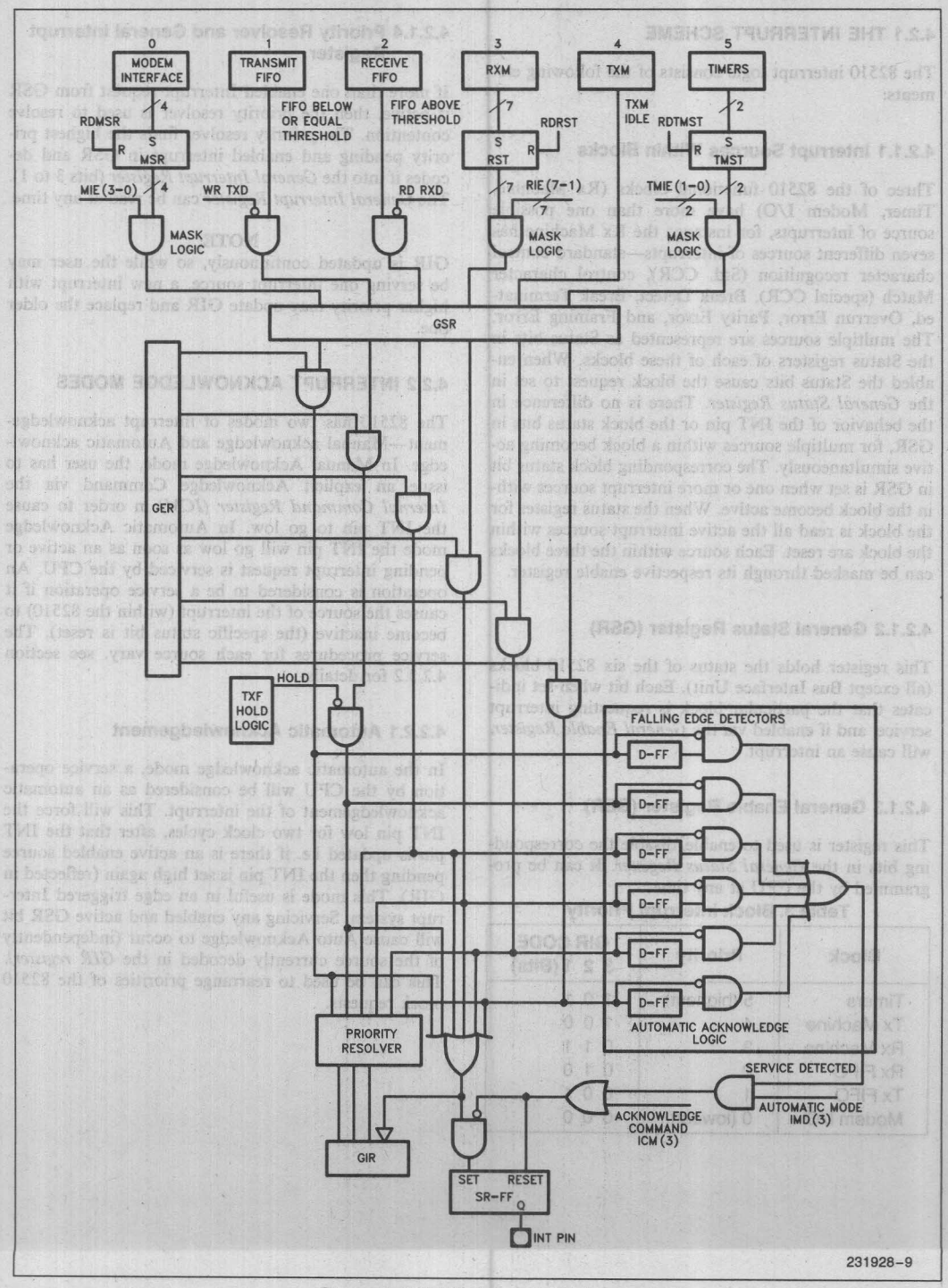


Figure 9. 82510's Interrupt Scheme

4.2.1 THE INTERRUPT SCHEME

The 82510 interrupt logic consists of the following elements:

4.2.1.1 Interrupt Sources Within Blocks

Three of the 82510 functional blocks (Rx Machine, Timer, Modem I/O) have more than one possible source of interrupts, for instance the Rx Machine has seven different sources of interrupts—standard control character recognition (Std. CCR), control character Match (special CCR), Break Detect, Break Terminated, Overrun Error, Parity Error, and Framing Error. The multiple sources are represented as Status bits in the Status registers of each of these blocks. When enabled the Status bits cause the block request to set in the *General Status Register*. There is no difference in the behavior of the INT pin or the block status bits in GSR, for multiple sources within a block becoming active simultaneously. The corresponding block status bit in GSR is set when one or more interrupt sources within the block become active. When the status register for the block is read all the active interrupt sources within the block are reset. Each source within the three blocks can be masked through its respective enable register.

4.2.1.2 General Status Register (GSR)

This register holds the status of the six 82510 blocks (all except Bus Interface Unit). Each bit when set indicates that the particular block is requesting interrupt service, and if enabled via the *General Enable Register,* will cause an interrupt.

4.2.1.3 General Enable Register (GER)

This register is used to enable/disable the corresponding bits in the *General Status Register*. It can be programmed by the CPU at any time.

Table 3. Block Interrupt Priority

Block	Priority	GIR CODE 3 2 1 (Bits)
Timers	5 (highest)	1 0 1
Tx Machine	4	1 0 0
Rx Machine	3	0 1 1
Rx FIFO	2	0 1 0
Tx FIFO	1	0 0 1
Modem I/O	0 (lowest)	0 0 0

4.2.1.4 Priority Resolver and General Interrupt Register

If more than one enabled Interrupt request from GSR is active, then the priority resolver is used to resolve contention. The priority resolver finds the highest priority pending and enabled interrupt in GSR and decodes it into the *General Interrupt Register* (bits 3 to 1). The *General Interrupt Register* can be read at any time.

NOTE:
GIR is updated continuously, so while the user may be serving one interrupt source, a new interrupt with higher priority may update GIR and replace the older one.

4.2.2 INTERRUPT ACKNOWLEDGE MODES

The 82510 has two modes of interrupt acknowledgement—Manual acknowledge and Automatic acknowledge. In Manual Acknowledge mode, the user has to issue an explicit Acknowledge Command via the *Internal Command Register (ICM)* in order to cause the INT pin to go low. In Automatic Acknowledge mode the INT pin will go low as soon as an active or pending interrupt request is serviced by the CPU. An operation is considered to be a service operation if it causes the source of the interrupt (within the 82510) to become inactive (the specific status bit is reset). The service procedures for each source vary, see section 4.2.3.2 for details.

4.2.2.1 Automatic Acknowledgement

In the automatic acknowledge mode, a service operation by the CPU will be considered as an automatic acknowledgement of the interrupt. This will force the INT pin low for two clock cycles, after that the INT pin is updated i.e. if there is an active enabled source pending then the INT pin is set high again (reflected in GIR). This mode is useful in an edge triggered Interrupt system. Servicing any enabled and active GSR bit will cause Auto Acknowledge to occur (independently of the source currently decoded in the *GIR register).* This can be used to rearrange priorities of the 82510 block requests.

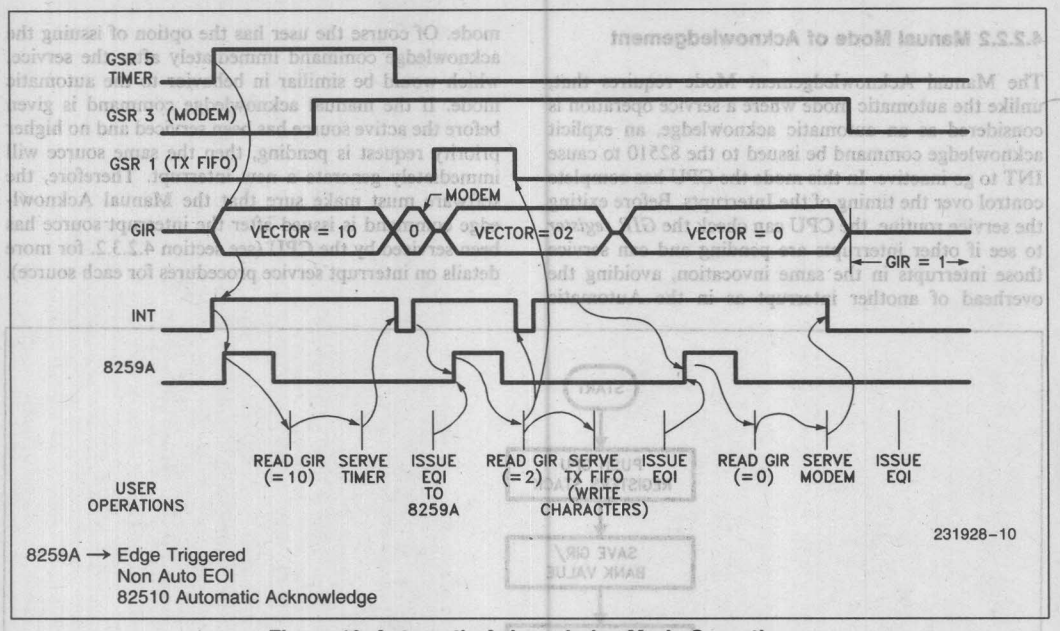
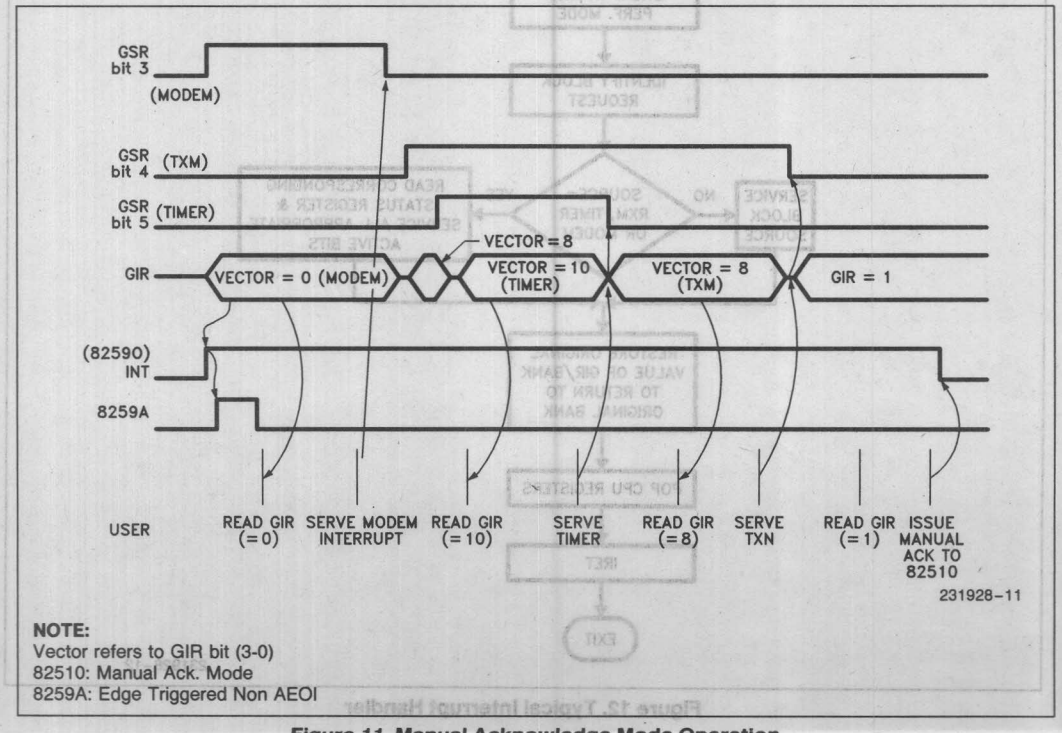


Figure 10. Automatic Acknowledge Mode Operation



NOTE:
Vector refers to GIR bit (3-0)
82510: Manual Ack. Mode
8259A: Edge Triggered Non AEOI

Figure 11. Manual Acknowledge Mode Operation

4.2.2.2 Manual Mode of Acknowledgement

The Manual Acknowledgement Mode requires that, unlike the automatic mode where a service operation is considered as an automatic acknowledge, an explicit acknowledge command be issued to the 82510 to cause INT to go inactive. In this mode the CPU has complete control over the timing of the Interrupts. Before exiting the service routine, the CPU can check the *GIR register* to see if other interrupts are pending and can service those interrupts in the same invocation, avoiding the overhead of another interrupt as in the Automatic mode. Of course the user has the option of issuing the acknowledge command immediately after the service, which would be similiar in behavior to the automatic mode. If the manual acknowledge command is given before the active source has been serviced and no higher priority request is pending, then the same source will immediately generate a new interrupt. Therefore, the software must make sure that the Manual Acknowledge command is issued after the interrupt source has been serviced by the CPU (see section 4.2.3.2. for more details on interrupt service procedures for each source).

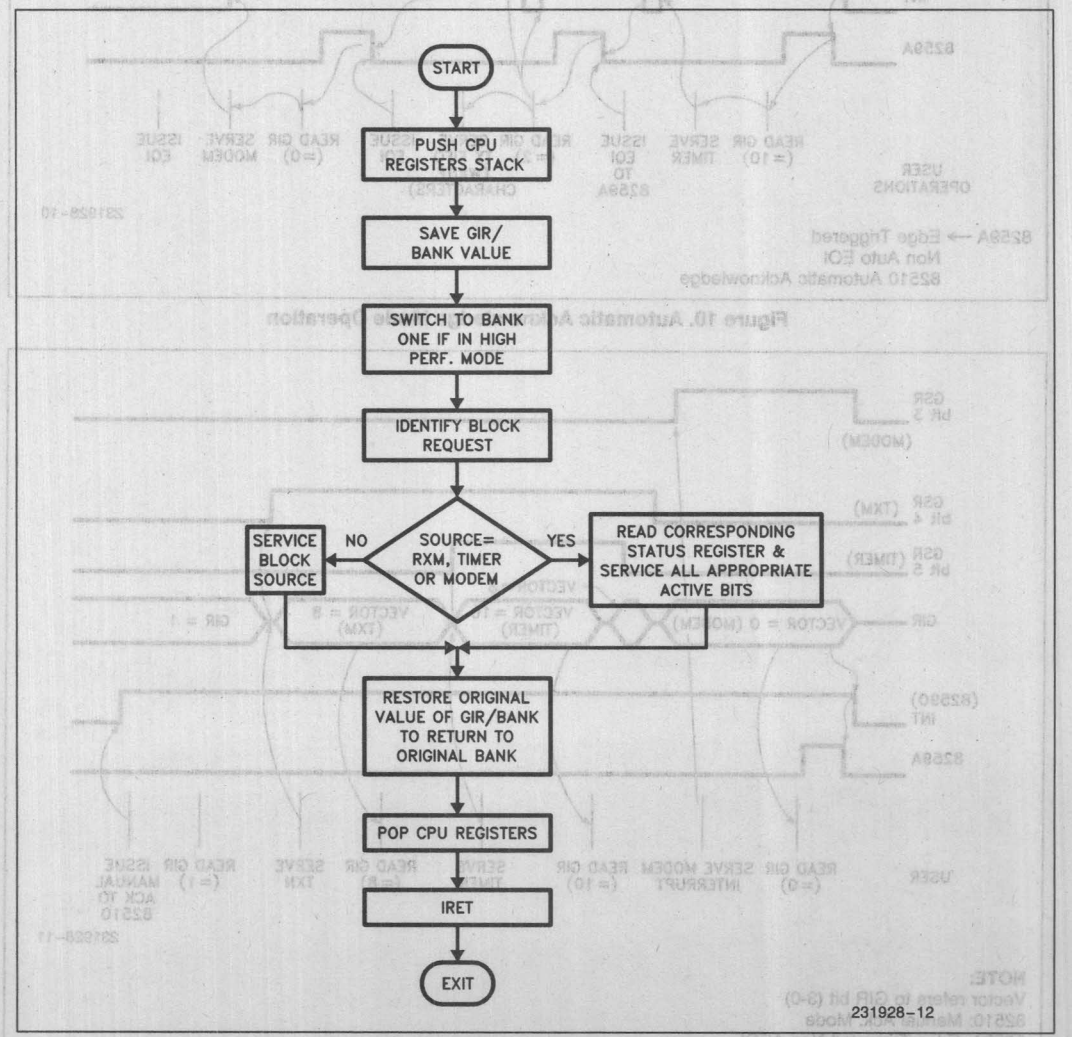


Figure 12. Typical Interrupt Handler

4.2.3 GENERAL INTERRUPT HANDLER

In general an interrupt handler for the 82510 must first identify the interrupt source within the 82510, transfer control to the appropriate service routine and then service the active source. The active source can be identified from two registers—*General Interrupt Register*, or *General Status Register*. The *GIR register* identifies the highest priority active block interrupt request. The *GSR register* identifies all active (pending or in service) Block Interrupt Requests. The typical operation of the 82510 interrupt handler is given in Figure 12. The two major issues of concern are the source identification and Control Transfer to the appropriate service routine.

Since the 82510 registers are divided into banks, and the interrupt handler may change register banks during service, it is best to save the bank being used by the main program and then do the interrupt processing. Upon completion of service, the original bank value is restored to the *GIR/Bank register.*

4.2.3.1 Source Identification

The 82510 has 16 interrupt sources, and the CPU must identify the source before performing any service. Although the procedure varies, the typical method would be to identify the block requesting service by reading

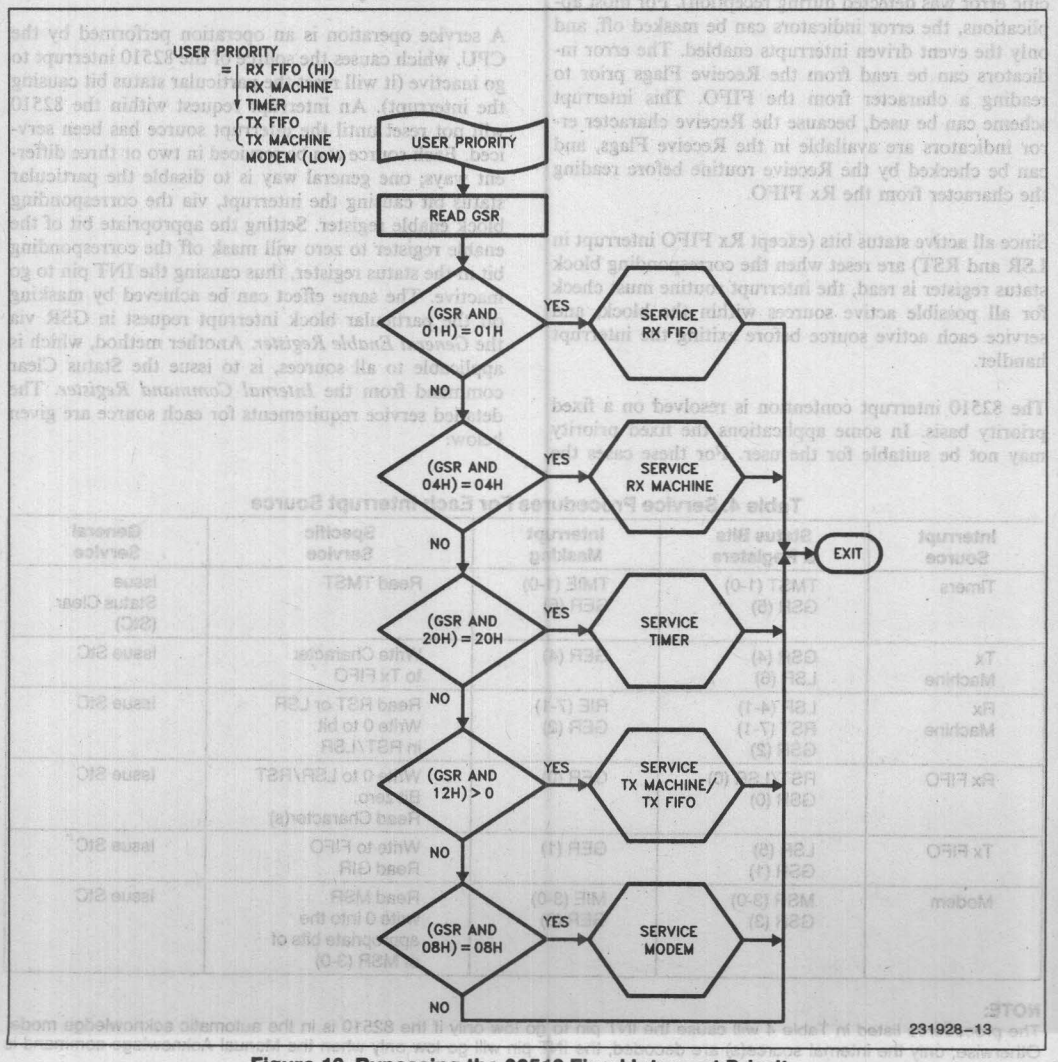


Figure 13. Bypassing the 82510 Fixed Interrupt Priority

GIR bits 3-1. If the source is either Tx Machine, Tx FIFO, or Rx FIFO, no further indentification is needed, the user can transfer control to the service routine (in most cases, only one Timer will be used, therefore the Timer Routine can also be directly invoked). All modem I/O interrupts can be handled via one routine as all the modem interrupt sources are supplementary to the modem handshaking function. The Rx Machine, however, has two different types of interrupt sources, event indications (CCR/Address recognition CCR/Address Match, Break Detect, Break Terminate, and Overrun Error), and error indications (Parity Error, Framing Error, these error indications do not refer to any particular character, they just indicate that the specific error was detected during reception). For most applications, the error indicators can be masked off, and only the event driven interrupts enabled. The error indicators can be read from the Receive Flags prior to reading a character from the FIFO. This interrupt scheme can be used, because the Receive character error indicators are available in the Receive Flags, and can be checked by the Receive routine before reading the character from the Rx FIFO.

Since all active status bits (except Rx FIFO interrupt in LSR and RST) are reset when the corresponding block status register is read, the interrupt routine must check for all possible active sources within the block, and service each active source before exiting the interrupt handler.

The 82510 interrupt contention is resolved on a fixed priority basis. In some applications the fixed priority may not be suitable for the user. For these cases the user can bypass the 82510's priority resolution by using the *General Status Register* (rather than GIR) to determine the block interrupt sources requesting service. Each source is checked in order of user priority and serviced when identified (There will be no problem with using this algorithm in auto acknowledge mode because the INT pin will go low as soon as a pending and enabled interrupt request goes low). The user will be trading some service latency time for additional source identification time, this algorithm's efficiency will improve as the number of block sources to verify is reduced. See Figure 13 for the algorithm.

4.2.3.2 Interrupt Service

A service operation is an operation performed by the CPU, which causes the source of the 82510 interrupt to go inactive (it will reset the particular status bit causing the interrupt). An interrupt request within the 82510 will not reset until the interrupt source has been serviced. Each source can be serviced in two or three different ways; one general way is to disable the particular status bit causing the interrupt, via the corresponding block enable register. Setting the appropriate bit of the enable register to zero will mask off the corresponding bit in the status register, thus causing the INT pin to go inactive. The same effect can be achieved by masking off the particular block interrupt request in GSR via the *General Enable Register.* Another method, which is applicable to all sources, is to issue the Status Clear command from the *Internal Command Register.* The detailed service requirements for each source are given below:

Table 4. Service Procedures For Each Interrupt Source

Interrupt Source	Status Bits & Registers	Interrupt Masking	Specific Service	General Service
Timers	TMST (1-0) GSR (5)	TMIE (1-0) GER (5)	Read TMST	Issue Status Clear (StC)
Tx Machine	GSR (4) LSR (6)	GER (4)	Write Character to Tx FIFO	Issue StC
Rx Machine	LSR (4-1) RST (7-1) GSR (2)	RIE (7-1) GER (2)	Read RST or LSR Write 0 to bit in RST/LSR	Issue StC
Rx FIFO	RST/LSR (0) GSR (0)	GER (0)	Write 0 to LSR/RST Bit zero. Read Character(s)	Issue StC
Tx FIFO	LSR (5) GSR (1)	GER (1)	Write to FIFO Read GIR	Issue StC
Modem	MSR (3-0) GSR (3)	MIE (3-0) GER (3)	Read MSR write 0 into the appropriate bits of of MSR (3-0)	Issue StC

NOTE:
The procedures listed in Table 4 will cause the INT pin to go low only if the 82510 is in the automatic acknowledge mode. Otherwise, only the internal source(s) are decoded, the INT pin will go low only when the Manual Acknowlege command is issued.

4.3 Polling

The 82510 can be used in a polling mode by using the *General Status Register* to determine the status of the various 82510 blocks, this is useful when the software must manage all the blocks at once. If the software is dedicated to performing one function at a time, then the specific status registers for the block can be used, e.g. if the software is only going to be Transmitting, it can monitor the Tx FIFO level by polling the *FIFO Level Register,* and write data whenever the Tx FIFO level decreases. Reception of data can be done in the same manner.

5.0 SOFTWARE CONSIDERATIONS

5.1 Configuration

The 82510 must be configured for the appropriate modes before it can be used to transmit or receive data. Configuration is done via read and write registers, each functional block (except for BIU) has a configuration register. Typically the configuration is done once after start up, however, the FIFO thresholds and the interrupt masks can be reconfigured dynamically. If the 82510 configuration is not known at start up it is best to bring the device to a known state by issuing a software reset command (ICM register, bank one). At this point all block interrupts are masked out in GER and all configuration and status registers have default values. The bank register is pointing to bank zero. The 82510 can now be configured as follows:

1. **If BRG A is being issued as a baud rate generator then load the baud rate count into BAL and BAH registers.**
2. **Configure the character attributes in LCR register (Parity, Stop Bit Length, and Character Length).**

(Note if interrupts are being used, steps 1 and 2 can also be done at the end, since the user will have to return to bank zero to set the interrupt masks in GER)

3. **Load ACR0 register with the appropriate Control or Address character (if using the Control Character Match or Address Match capability of the 82510).**
4. **Switch to Bank two.**

   (In this Bank the configuration can be done in any order)
5. **Configure the Receive and Transmit FIFO thresholds if using different thresholds than the default).**
6. **Configure the Transmit Mode Register for the Stop Bit length, modem control, and if using echo or 9 bit length or software parity, configure the appropriate bits of the register. The default mode of the modem control is Manual, if using the FIFO then the automatic mode would be most useful).**
7. **Configure the Rx FIFO depth, interrupt acknowledge mode, μlan or normal mode and echo modes in IMD register.**
8. **Load ACR1 if necessary**
9. **Enable Rx Machine Interrupts as necessary via RIE.**
10. **Configure RMD for CCR, DPLL operation, Sampling Window, and start bit.**
11. **Switch to Bank 3.**
12. **Configure CLCF register for Tx and Rx clocks and or Sources**
13. **Configure BACF register for BRG A mode and source.**
14. **Load BBL and BBH if BRGB is being used (as either a BRG or a Timer).**
15. **Configure BBCF register if necessary.**
16. **If reconfiguration of the modem pin is necessary then program the PMD register.**
17. **Enable any modem interrupt sources, if required, via MIE register.**
18. **Enable Timer interrupts, if necessary, via TMIE.**
19. **If using interrupts**

    then

    i) **Switch to Bank zero.**

    Disable Interrupts at CPU (either by masking the request at the interrupt controller or executing the CLI instruction).

    ii) **Enable the appropriate 82510 Block interrupts by setting bits in the GER register. (CPU interrupts can now be reenabled, but it is recommended to switch banks before enabling the CPU interrupts).**

NOTE:
At this stage it is best to leave the TxM and Tx FIFO interrupt disabled. See section 6.3 Transmit Operation for details)

20. **Switch to Bank One. Load Transmit Flags if using 9-bit characters, or 8051 9-bit mode or software parity. If using interrupts CPU interrupts can now be enabled.**

Bank One is used for general operation, the 82510 can now be used to transmit or receive characters.

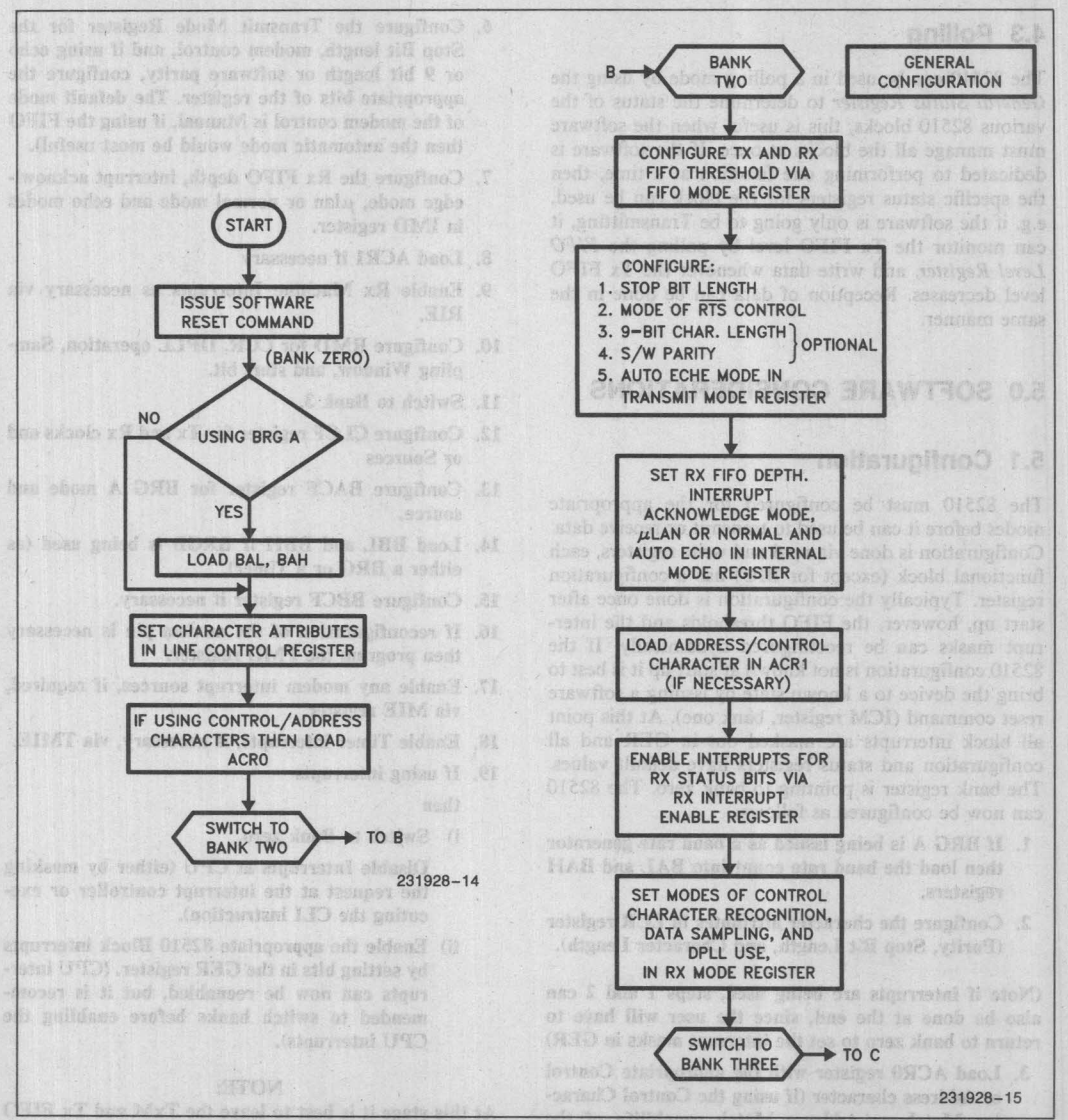


Figure 14. Configuration Flow Chart

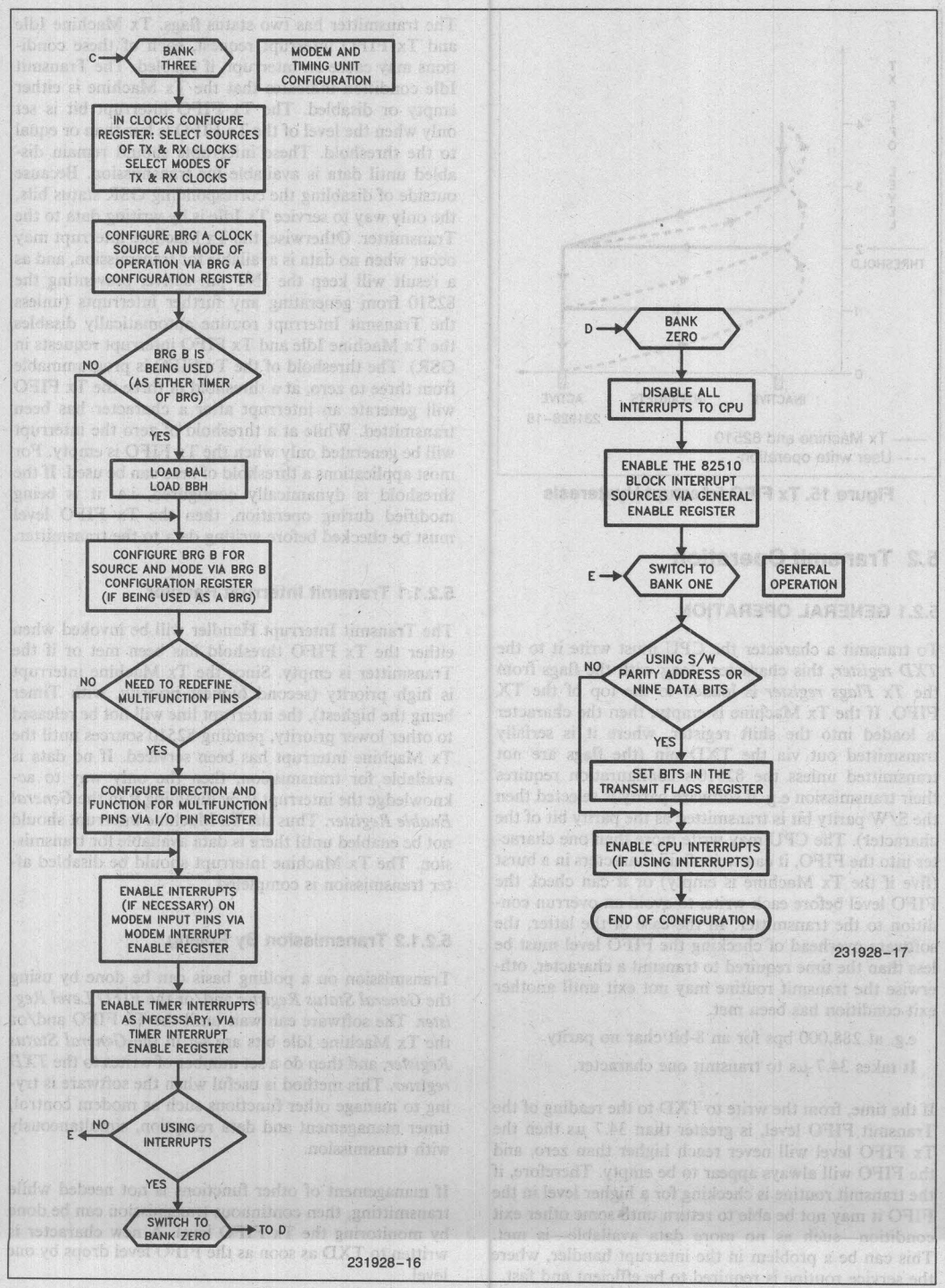


Figure 14. Configuration Flow Chart (Continued)

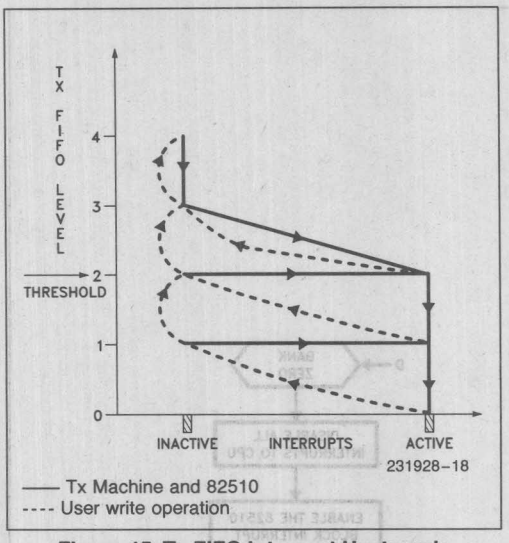


Figure 15. Tx FIFO Interrupt Hysteresis

5.2 Transmit Operation

5.2.1 GENERAL OPERATION

To transmit a character the CPU must write it to the *TXD register*, this character along with the flags from the *Tx Flags register* is loaded to the top of the TX FIFO. If the Tx Machine is empty, then the character is loaded into the shift register, where it is serially transmitted out via the TXD pin (the flags are not transmitted unless the 82510's configuration requires their transmission e.g. if software parity is selected then the S/W parity bit is transmitted as the parity bit of the character). The CPU may write more than one character into the FIFO, it can write four characters in a burst (five if the Tx Machine is empty) or it can check the FIFO level before each write, to avoid an overrun condition to the transmitter. In the case of the latter, the software overhead of checking the FIFO level must be less than the time required to transmit a character, otherwise the transmit routine may not exit until another exit condition has been met.

e.g. at 288,000 bps for an 8-bit char no parity

It takes 34.7 μs to transmit one character.

If the time, from the write to TXD to the reading of the Transmit FIFO level, is greater than 34.7 μs then the Tx FIFO level will never reach higher than zero, and the FIFO will always appear to be empty. Therefore, if the transmit routine is checking for a higher level in the FIFO it may not be able to return until some other exit condition—such as no more data available—is met. This can be a problem in the interrupt handler, where the service routine is required to be efficient and fast.

The transmitter has two status flags. Tx Machine Idle and Tx FIFO interrupt request, each of these conditions may cause an interrupt, if enabled. The Transmit Idle condition indicates that the Tx Machine is either empty or disabled. The Tx FIFO interrupt bit is set only when the level of the Tx FIFO is less than or equal to the threshold. These interrupts should remain disabled until data is available for transmission. Because outside of disabling the corresponding GSR status bits, the only way to service Tx Idle is by writing data to the Transmitter. Otherwise, the Tx Machine interrupt may occur when no data is available for transmission, and as a result will keep the INT pin active, preventing the 82510 from generating any further interrupts (unless the Transmit Interrupt routine automatically disables the Tx Machine Idle and Tx FIFO interrupt requests in GSR). The threshold of the Tx FIFO is programmable from three to zero, at a threshold of three the Tx FIFO will generate an interrupt after a character has been transmitted. While at a threshold of zero the interrupt will be generated only when the Tx FIFO is empty. For most applications a threshold of zero can be used. If the threshold is dynamically configured, i.e. it is being modified during operation, then the Tx FIFO level must be checked before writing data to the transmitter.

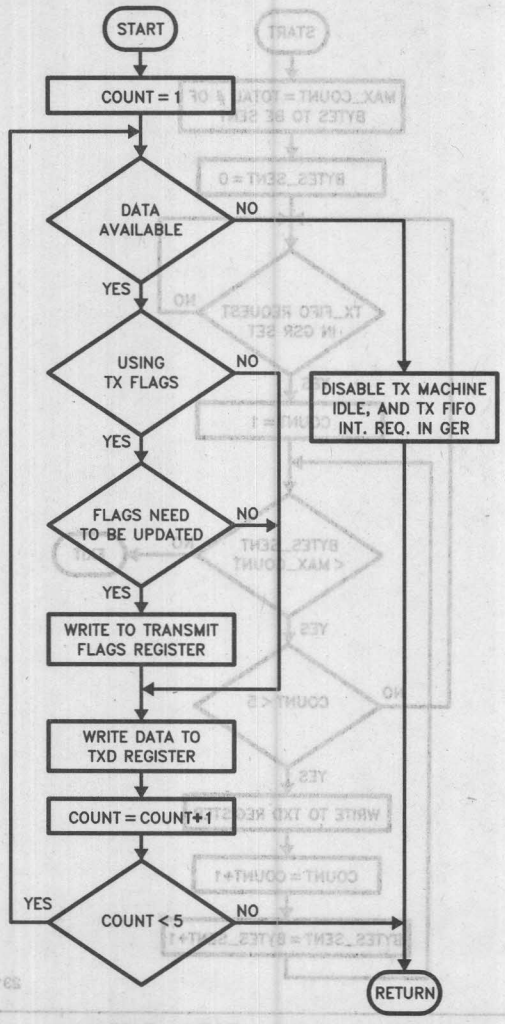
5.2.1.1 Transmit Interrupt Handler

The Transmit Interrupt Handler will be invoked when either the Tx FIFO threshold has been met or if the Transmitter is empty. Since the Tx Machine interrupt is high priority (second highest priority, with Timer being the highest), the interrupt line will not be released to other lower priority, pending 82510 sources until the Tx Machine interrupt has been serviced. If no data is available for transmission, then the only way to acknowledge the interrupt is by disabling it in the *General Enable Register.* Thus the Tx Machine interrupt should not be enabled until there is data available for transmission. The Tx Machine interrupt should be disabled after transmission is completed.

5.2.1.2 Transmission By Polling

Transmission on a polling basis can be done by using the *General Status Register* and/or the *FIFO Level Register.* The software can wait until the Tx FIFO and/or the Tx Machine Idle bits are set in the *General Status Register*, and then do a set number of writes to the *TXD register.* This method is useful when the software is trying to manage other functions such as modem control, timer management and data reception, simultaneously with transmission.

If management of other functions is not needed while transmitting, then continuous transmission can be done by monitoring the Tx FIFO level. A new character is written to TXD as soon as the FIFO level drops by one level.



Tx FIFO Threshold = 0

231928-19

NOTE:
TxM Idle and Tx FIFO Empty interrupts are enabled by the Main Program, when data transmission is required.

Figure 16. 16 Tx Interrupt Handler Flow Chart

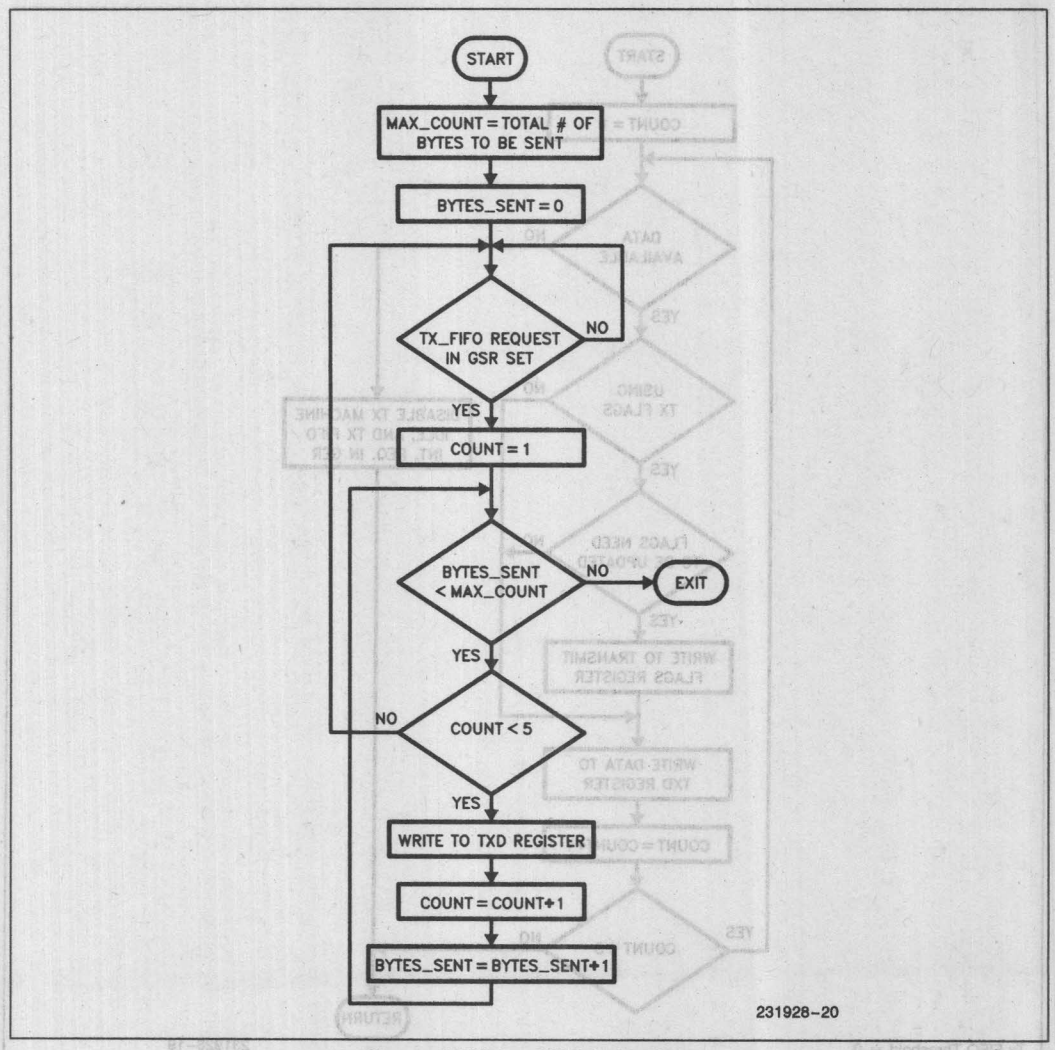


Figure 17. Using GSR for Polling

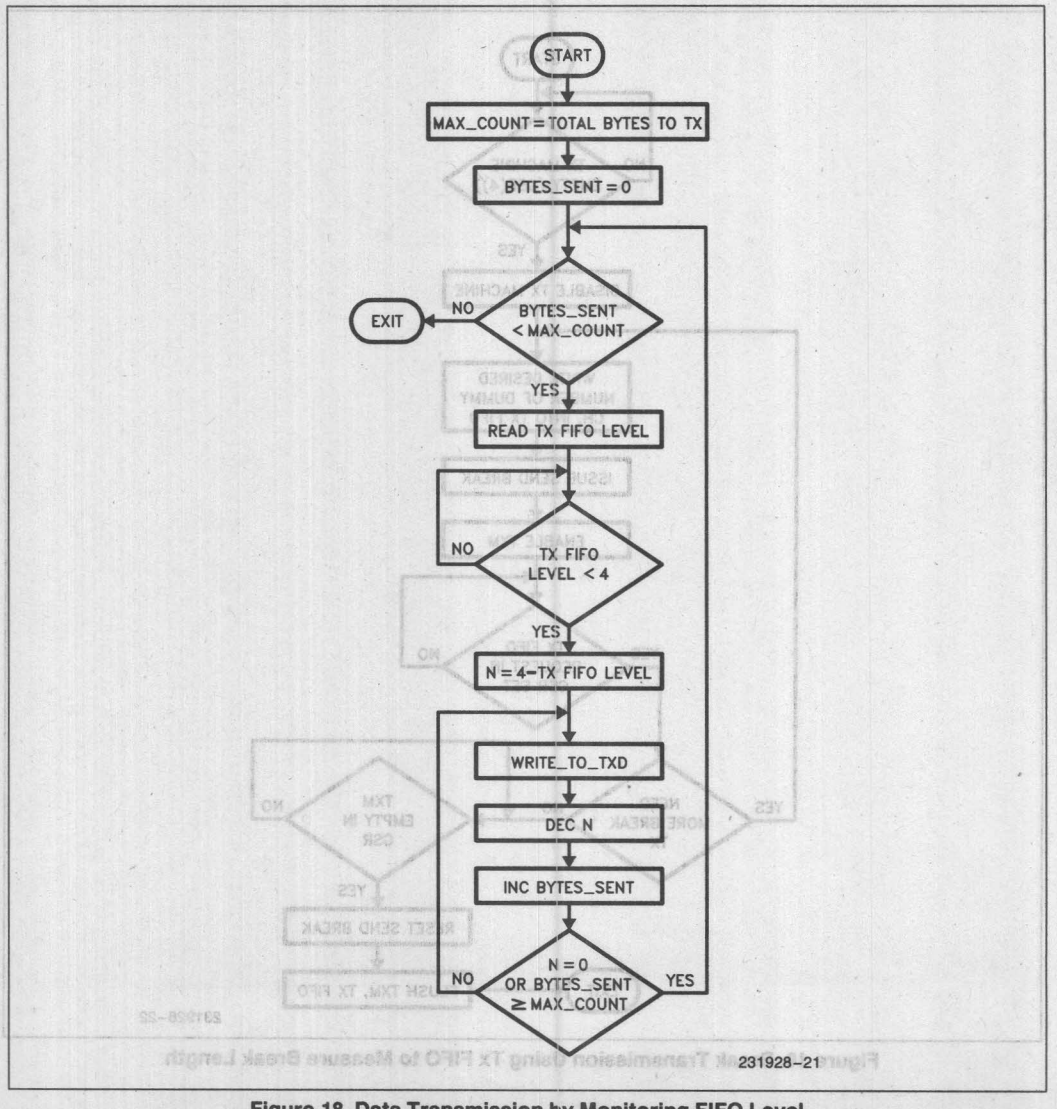


Figure 18. Data Transmission by Monitoring FIFO Level

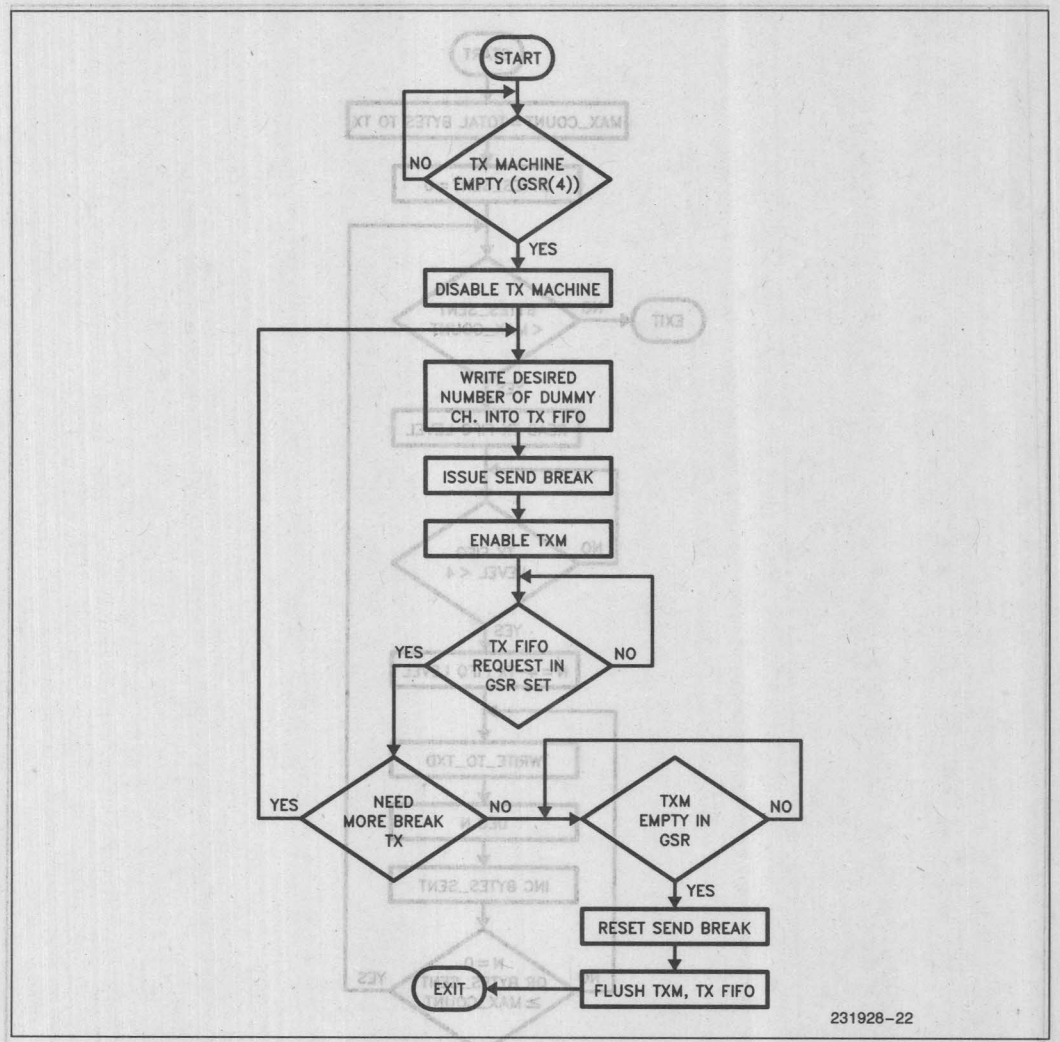
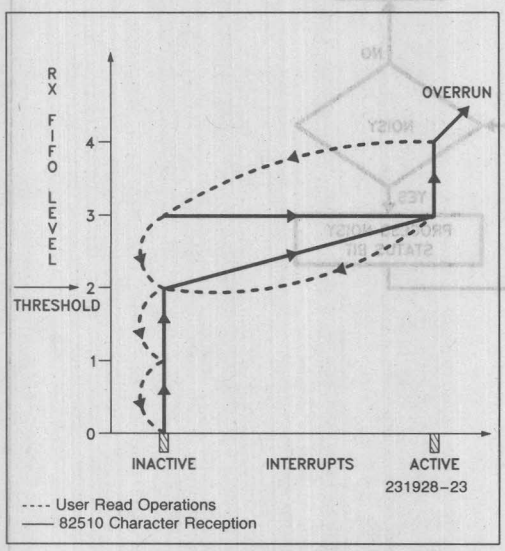


Figure 19. Break Transmission Using Tx FIFO to Measure Break Length

5.2.1.3 Break Transmission

The 82510 will transmit a break when bit six of the *Line Control Register* is set high. This will cause the TXD pin to be held at Mark for one or more character time. The Tx FIFO can be used to program a variable length break, see Figure 19 for details. If the break command is issued in the midst of character transmission the TXD pin will go low, but the transmitter will not be disabled. The characters from the Tx FIFO will be shifted out on to the Tx Machine and lost. To prevent the erroneous transmission of data, The CPU must make sure the Transmitter is empty or disabled before issuing the Send Break command.



Rx FIFO Hysteresis

5.3 Data Reception

The receiver provides the 82510 with three types of information:

a) Data characters received

b) Rx Flags for each data character

c) Status information on events within the Rx Machine.

The Rx FIFO interrupt request goes active when the Rx FIFO level is greater than the threshold, if the interrupt for this bit is enabled then it will generate an interrupt to the CPU. This is a request for the CPU to read characters from the 82510. Each character on the Rx FIFO has flags associated with it, all of these flags are generated by the Rx Machine during reception of the character. These flags provide information on the integrity of the character, e.g. whether the character was received OK, or if there were any errors. The receiver status is provided via the *Receive Status Register (RST),* which provides information on events occurring within the Rx Machine, since the last time RST was read. The information may or may not apply to the current character being read from the *RXD register.* The CPU may read one or more characters from the Rx FIFO. After each read, if the FIFO contains more than a single character, a new character is loaded into the *RXD register* and the flags for that character are placed into the *RXF register.* The software can check for the Rx character OK bit in the flags to make sure that the character was received without any problems.

2

5.3.1 RECEIVE INTERRUPT HANDLER

The Receiver will generate two types of interrupts, Rx FIFO interrupt and Rx Machine Interrupt. The Rx FIFO interrupt requires that the CPU read data characters from the Rx FIFO. If the Rx Machine interrupts are disabled then the CPU should also check for errors in the character before moving it to a valid buffer. The interrupts generated by the Rx Machine can be divided into two categories—occurrence of errors during reception of data (parity error, framing error, overrun error), or the occurrence of certain events (Control/Address character received, Break detected, Break Terminated). For typical applications, the error status of each received character can be checked via the *Receive Flags,* and the events can be handled via interrupts.

5.3.2 RECEIVING DATA BY POLLING

To receive data through polling, the 82510 can use the *General Status or the Receive Status Registers* to check for the Rx FIFO request. If the Receive routine does not generate time outs or modem pin transitions, then the data can also be received by monitoring the Rx FIFO level in the *FIFO Level Register.* The implementation using GSR would be useful in applications where the software routine must monitor the timer for time outs or the modem pins for change in status. The example polling routine illustrates the use of the *FIFO Level Register* in receiving data. It waits for the Rx FIFO request before beginning data reception. The procedure Rx__Data__Poll will receive the number of characters requested in Char__count and place them in the Receive buffer.

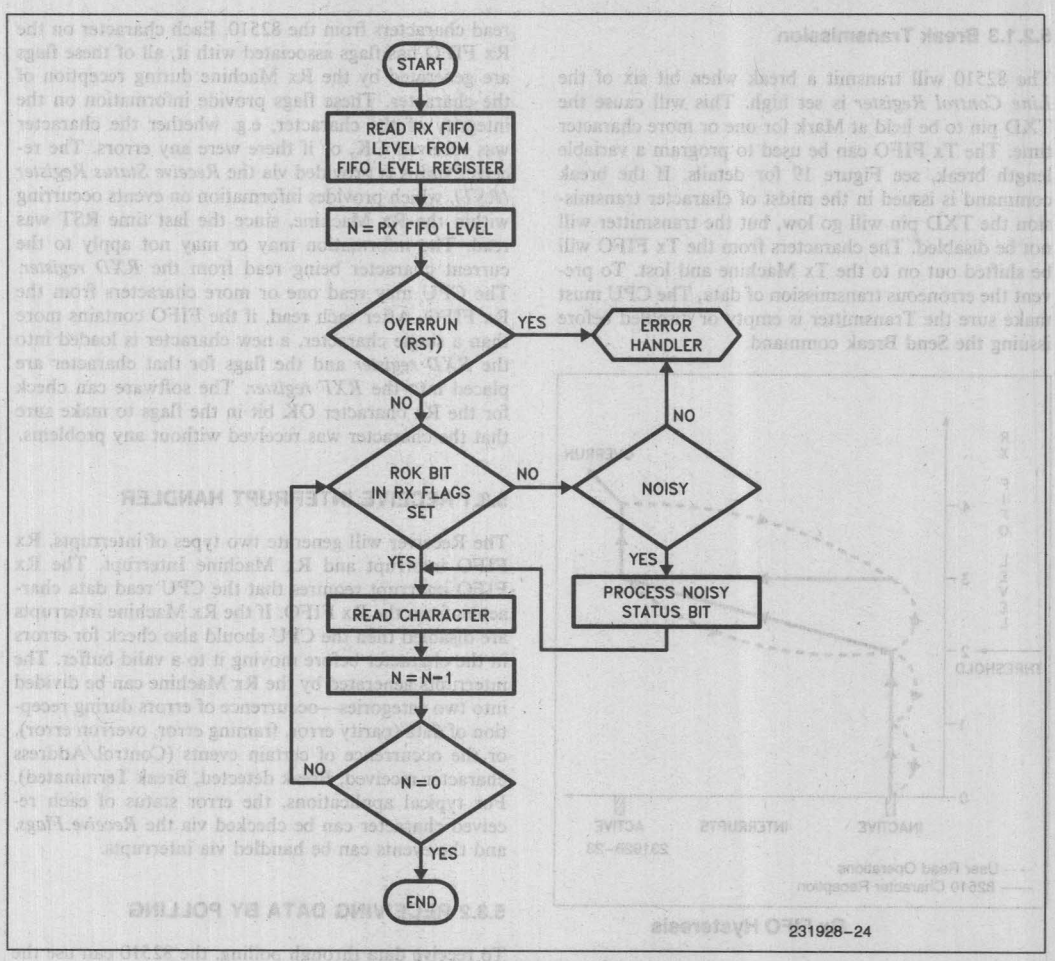


Figure 20. Rx FIFO Interrupt Handler

```
#define base 0x3F8;      /* base address of 82510 */
#define buff__size 128;

Rx__Data__Poll (Char__count, Rxbuffer)
int Char__count;         /* Total # of bytes to be received */
char *Rxbuffer [buffsize];
{
int count = 0;
int status, IvI, Rok;

While (((status = (Inp(base+7) & 0x05)) == 0x01) /* If Rx FIFO Req in GSR set */
{                                                /* Assume in bank one */
   /* If Rx FIFO is not empty */
   While ((IvI = ((Inp (base+4) & 0x70)/0x10)0&&(count < (Char__count))
   {
      /* If Character Received OK */
      if (((Rok = (Inp (base+1) & 0x60)) == 0x40)
      {
         Rxbuffer [count] = Inp (base);
         ++count;
      {
   {
{
{
```

Figure 21. Example Polling Routine

CONTROL CHARACTER HANDLING

The 82510 has two modes of control character recognition. It can recognize either standard ASCII or standard EBCDIC control characters, or it can recognize a match with two user programmed control (or Address Characters in MCS-51 9-bit mode, for Automatic Wake up) characters. Each mode generates an interrupt through the *Receive Status Register.* The *Receive Flags* also indicate whether the character being read is a control character. The usage of CCR depends on the maximum number of possible control characters that can be received at any one time. Applications such as Terminal Drivers, which have no more than two control characters outstanding, such as XON and Ctl-C, or XOFF and Ctl-C, can use just the Control Character Match mode by programming the registers ACR0 and ACR1. If the CPU needs to process text on a line by line basis, the standard Control Character recognition capability can be used to determine when an end of line has occurred e.g. a whole line has been received when a Carriage Return (CR) or Line Feed (LF) is received by the UART.

Implementation of a character oriented asynchronous file transfer protocol can be done using both standard and specific Control Character Recognition. In such protocols most control characters such as Start of Header (SOH), can only be received during certain states, these characters can be received via Standard *Control Character Recognition. A few Control Characters* (e.g. abort) can be received at any stage of communication, these can be received by using the Control Character Matching capabilities of the 82510.

5.3.3 BREAK RECEPTION

The 82510 has two status indications of break reception, Break Detect indicates that a break has been detected on the RXD pin. Break Terminated indicates that the Break previously detected on the RXD line has terminated and normal Data reception can resume. Each of these status bits can generate an interrupt request through the Rx Machine Interrupt request. Normal consequence of break is to abort the data reception or to introduce a line idle delay in the middle of data reception. In the case of the former, the Break Detect interrupt can be used to reset the 82510 Receive Machine and the Rx routine flags; in the case of the latter, the break terminated interrupt can be used to filter out the break characters and resume normal reception. Each break character is identified by a break flag in the *Rx Flags Register* (the CCR flag, Framing error, and CCR Match flag also may become active when a break character is received) and is loaded onto the Rx FIFO as a NULL character. If break continues even after the Rx FIFO is full, then an overrun error will occur but no further break characters will be loaded on to the Rx FIFO. The user can also measure the length of the break character stream by using the Timer.

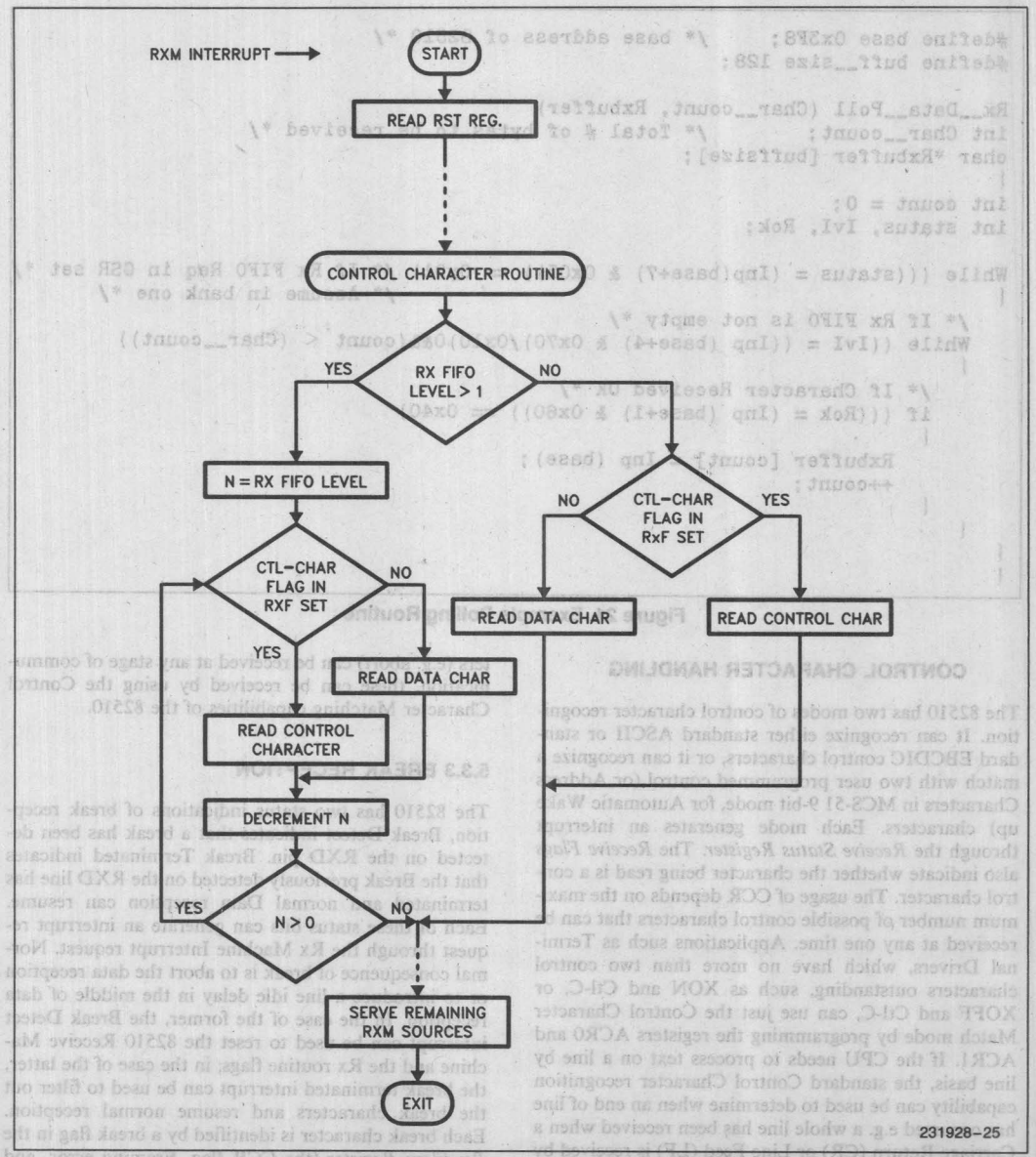


Figure 22. Handling Control Character Interrupts

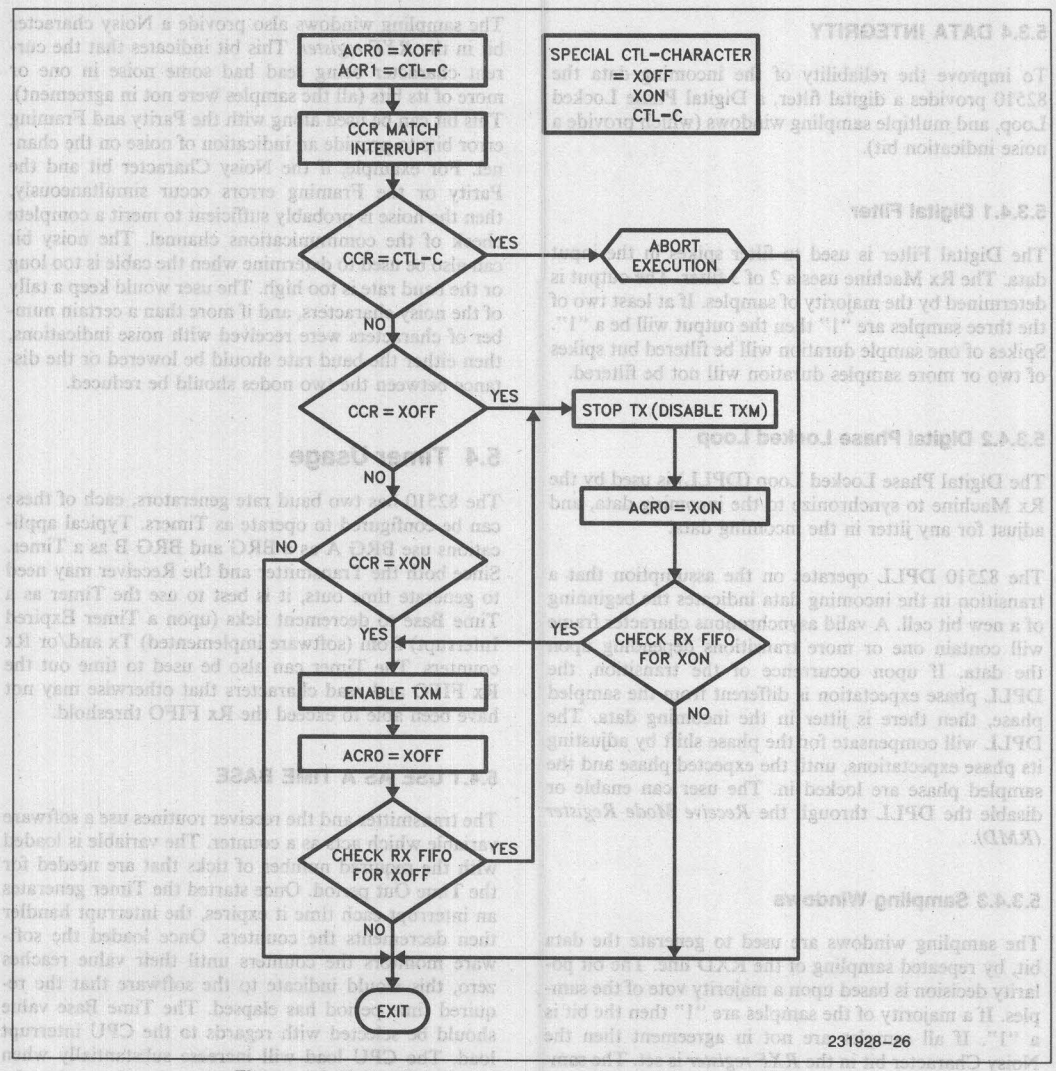


Figure 23. Using Control Character Match in Terminal Ports

5.3.4 DATA INTEGRITY

To improve the reliability of the incoming data the 82510 provides a digital filter, a Digital Phase Locked Loop, and multiple sampling windows (which provide a noise indication bit).

5.3.4.1 Digital Filter

The Digital Filter is used to filter spikes in the input data. The Rx Machine uses a 2 of 3 filter. The output is determined by the majority of samples. If at least two of the three samples are "1" then the output will be a "1". Spikes of one sample duration will be filtered but spikes of two or more samples duration will not be filtered.

5.3.4.2 Digital Phase Locked Loop

The Digital Phase Locked Loop (DPLL) is used by the Rx Machine to synchronize to the incoming data, and adjust for any jitter in the incoming data.

The 82510 DPLL operates on the assumption that a transition in the incoming data indicates the beginning of a new bit cell. A valid asynchronous character frame will contain one or more transitions depending upon the data. If upon occurrence of the transition, the DPLL phase expectation is different from the sampled phase, then there is jitter in the incoming data. The DPLL will compensate for the phase shift by adjusting its phase expectations, until the expected phase and the sampled phase are locked in. The user can enable or disable the DPLL through the *Receive Mode Register (RMD).*

5.3.4.3 Sampling Windows

The sampling windows are used to generate the data bit, by repeated sampling of the RXD line. The bit polarity decision is based upon a majority vote of the samples. If a majority of the samples are "1" then the bit is a "1". If all samples are not in agreement then the Noisy Character bit in the *RXF register* is set. The sampling windows are programmable for either 3 of 16 or 7 of 16. The 3/16 mode improves the jitter tolerance of the medium. While the 7/16 window improves the impulse noise tolerance of the channel.

The sampling windows also provide a Noisy character bit in the *RXF register.* This bit indicates that the current character being read had some noise in one or more of its bits (all the samples were not in agreement). This bit can be used along with the Parity and Framing error bits to provide an indication of noise on the channel. For example, if the Noisy Character bit and the Parity or the Framing errors occur simultaneously, then the noise is probably sufficient to merit a complete check of the communications channel. The noisy bit can also be used to determine when the cable is too long or the baud rate is too high. The user would keep a tally of the noisy characters, and if more than a certain number of characters were received with noise indications, then either the baud rate should be lowered or the distance between the two nodes should be reduced.

5.4 Timer Usage

The 82510 has two baud rate generators, each of these can be configured to operate as Timers. Typical applications use BRG A as a BRG and BRG B as a Timer. Since both the Transmitter and the Receiver may need to generate time outs, it is best to use the Timer as a Time Base to decrement ticks (upon a Timer Expired Interrupt) from (software implemented) Tx and/or Rx counters. The Timer can also be used to time out the Rx FIFO and read characters that otherwise may not have been able to exceed the Rx FIFO threshold.

5.4.1 USE AS A TIME BASE

The transmitter and the receiver routines use a software variable which acts as a counter. The variable is loaded with the required number of ticks that are needed for the Time Out period. Once started the Timer generates an interrupt each time it expires, the interrupt handler then decrements the counters. Once loaded the software monitors the counters until their value reaches zero, this would indicate to the software that the required time period has elapsed. The Time Base value should be selected with regards to the CPU interrupt load. The CPU load will increase substantially when the Timer is used as a Time Base, therefore using the Timer in this mode at very high baud rates may cause character overruns. A time base of 5 or 1 ms is probably the most useful. An additional benefit of the Time Base is that it can support more than two counters if required.

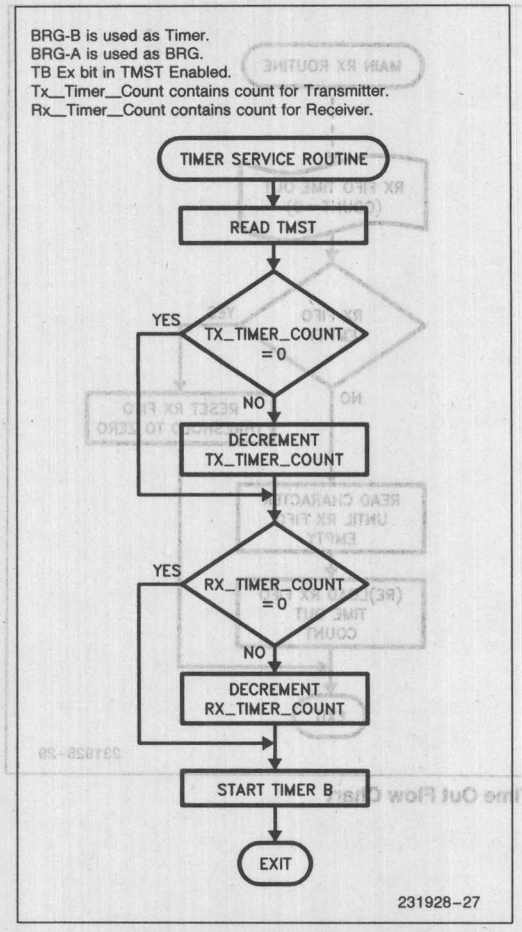


Figure 24. Timer use as Time Base for Transmit and Receive

5.4.2 USE FOR RX FIFO TIME OUT

In the 82510, Rx FIFO interrupts will occur only after the FIFO level has exceeded the threshold. Due to this mechanism and the nonuniform arrival rate of characters in asynchronous communications, there is a chance that characters will be "trapped" in the Rx FIFO for an extended period of time.

For example, assume the 82510 is a serial port on a system and is connected to a terminal. The user is entering a command line. The Rx FIFO Threshold = 3, and at the end only two bytes are received. Since the FIFO threshold has not been exceeded, the Rx FIFO interrupt is not generated. No other characters are received for 30 minutes, if the characters (in the Rx FIFO) are a line feed and carriage return, respectively, the CPU may be waiting for the CR to process the characters it has received. Consequently the characters will not be processed for 30 minutes.

In order to avoid such situations, a Rx FIFO Time Out mechanism can be implemented by using the 82510 Timer. The time out indicates that a certain amount of time has elapsed since the last read operation was performed. It causes the CPU to check the Rx FIFO and read any characters that are present.

In applications where the character reception occurs in a spurious manner (the exact number of characters cannot be guaranteed), the Rx FIFO Time Out is the only way to prevent characters from being trapped. The time out period is measured from the last read operation, every read operation resets the Rx FIFO Timer. To synchronize with the beginning of the data reception, initially the Rx FIFO threshold is set to zero. After the first character has been received, the threshold is adjusted to the desired value. When a Rx FIFO time out occurs and no data is available, the threshold is reset to zero. In error free data transmission, the beginning of data transmission is signaled by the reception of a control character, such as SOH or STX, the Rx FIFO time out mechanism should be triggered to the reception of these control characters.

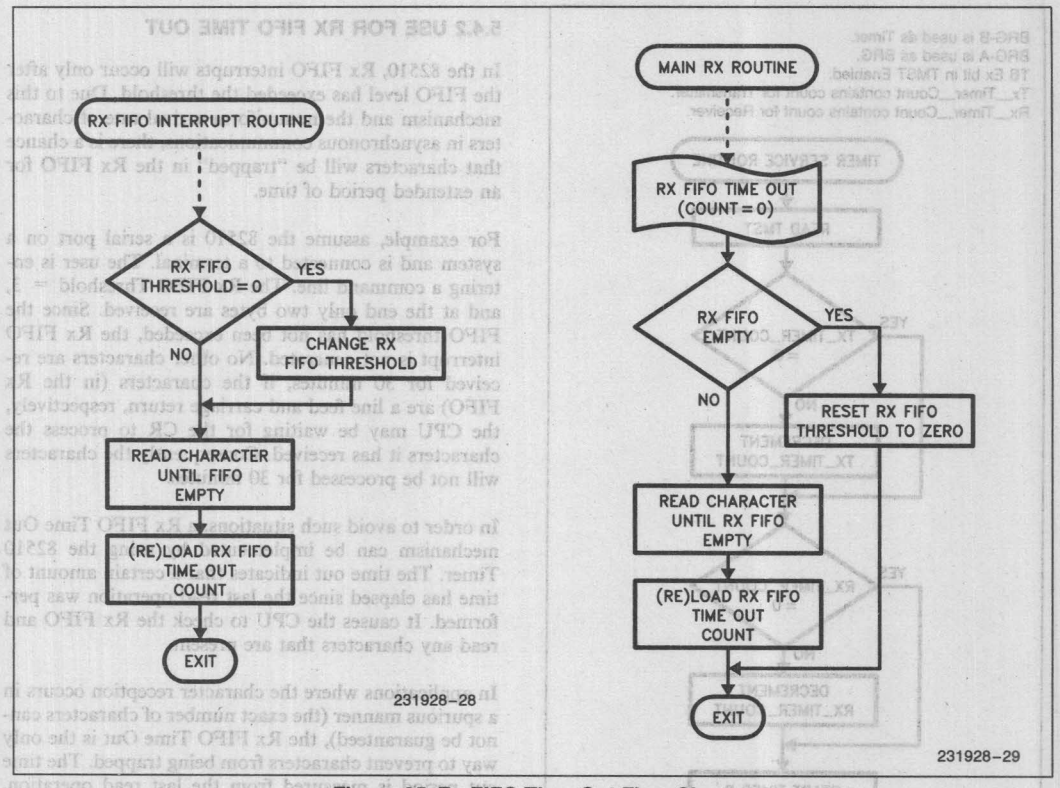


Figure 25. Rx FIFO Time Out Flow Chart

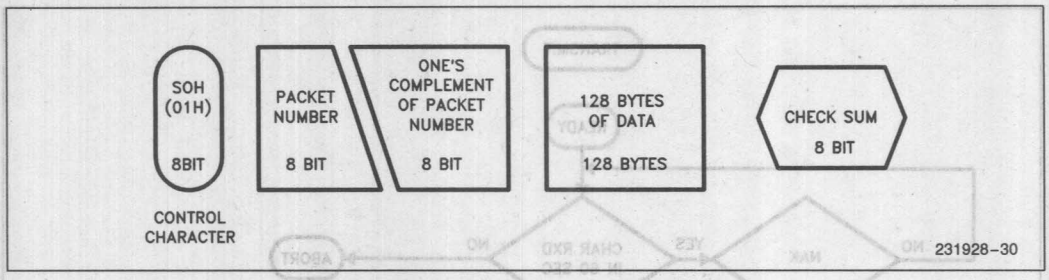


Figure 26. Packet Structure of XMODEM

6.0 82510 IMPLEMENTATION OF XMODEM

The 82510 XMODEM implementation is a file transfer program for the 82510 based on the XMODEM protocol. The software runs on the PC AT on a especially designed adapter board (the adapter board design is shown in Figure 33). The software uses most of the 82510 features including the baud rate generator, Timer, Control Character Recognition and FIFOs. The software uses an interrupt driven implementation, written in both assembly and C languages.

6.1 XMODEM Protocol

XMODEM is a popular error free data transfer protocol for asynchronous communications. Data is transferred in fixed length 128 byte packets, each packet has a checksum for error checking. The packets are delineated by control characters, which act as flags between the Receiver and the Transmitter. There are four control characters, SOH, EOT, ACK, and NAK. SOH indicates the Start of a Packet, EOT indicates the End Of Transmission; ACK and NAK are positive or negative acknowledgements of the packet respectively. The packet structure and protocol flow of XMODEM is provided in the figures given below.

6.2 Software

Interrupts are used to transmit and receive data. The software is implemented as two independent finite state machines—Transmit State Machine and Receive State Machine. Each state machine is triggered by external events such as user commands and data or Control Character reception. The state machines communicate with the 82510 interrupt service routines through software flags. The overall structure of the main routine is given in Figure 31. The major modules of the software are given in the hierarchy Chart, Figure 34, which lists the different modules in order.

The interface between the main program and the interrupt service routine is done through global flags. The interrupt handler services four sources—Transmit, Timer, Receive, and Control Characters. Each of the interrupt sources communicates with each of the state machines through the global flags. The state machines keep track of their individual states through state variables. The interface between the individual states within a state machine is done through state flags. The state machine diagrams are given in Figure 29 and Figure 30.

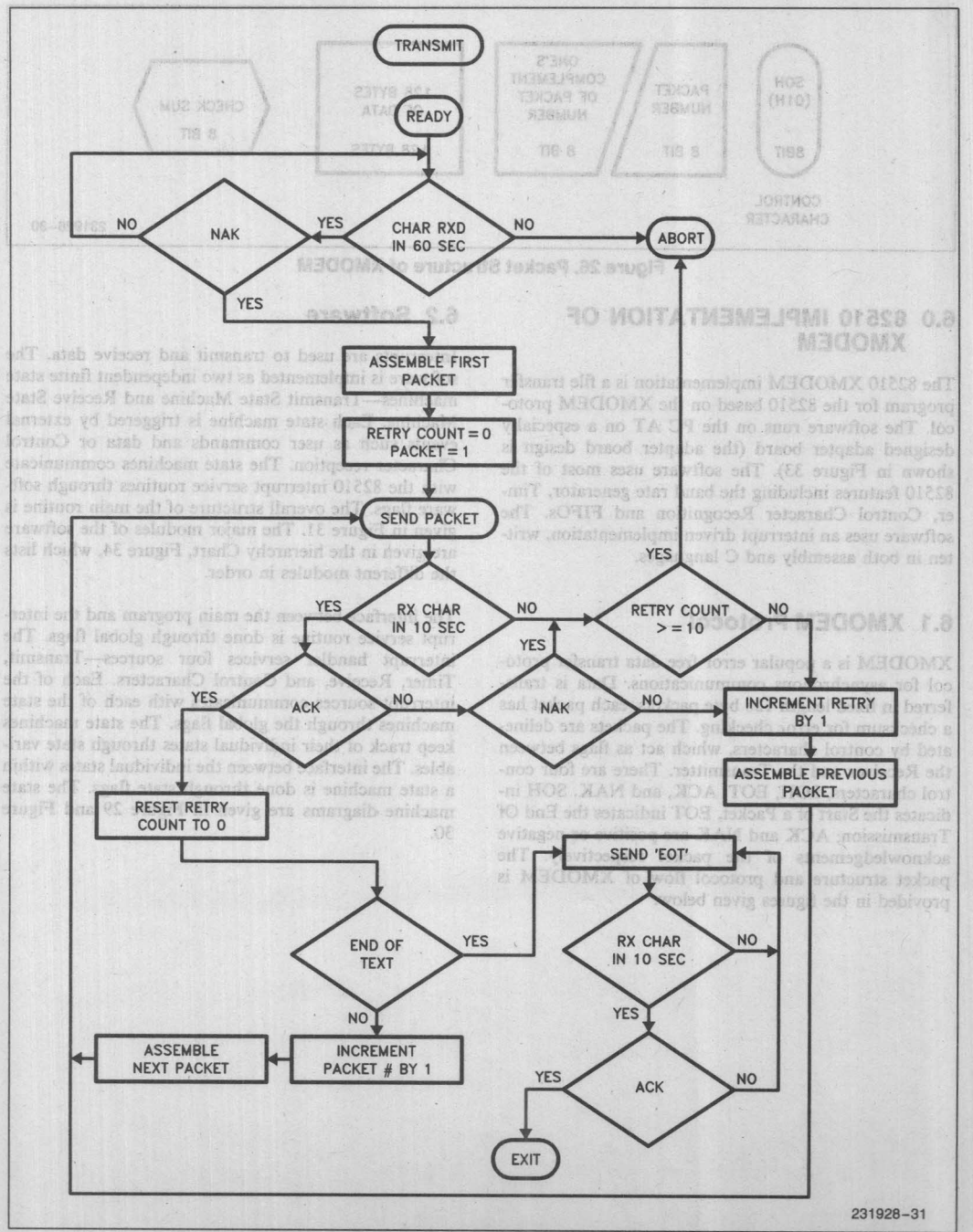


Figure 27. Protocol Flow for Transmit Side of XMODEM

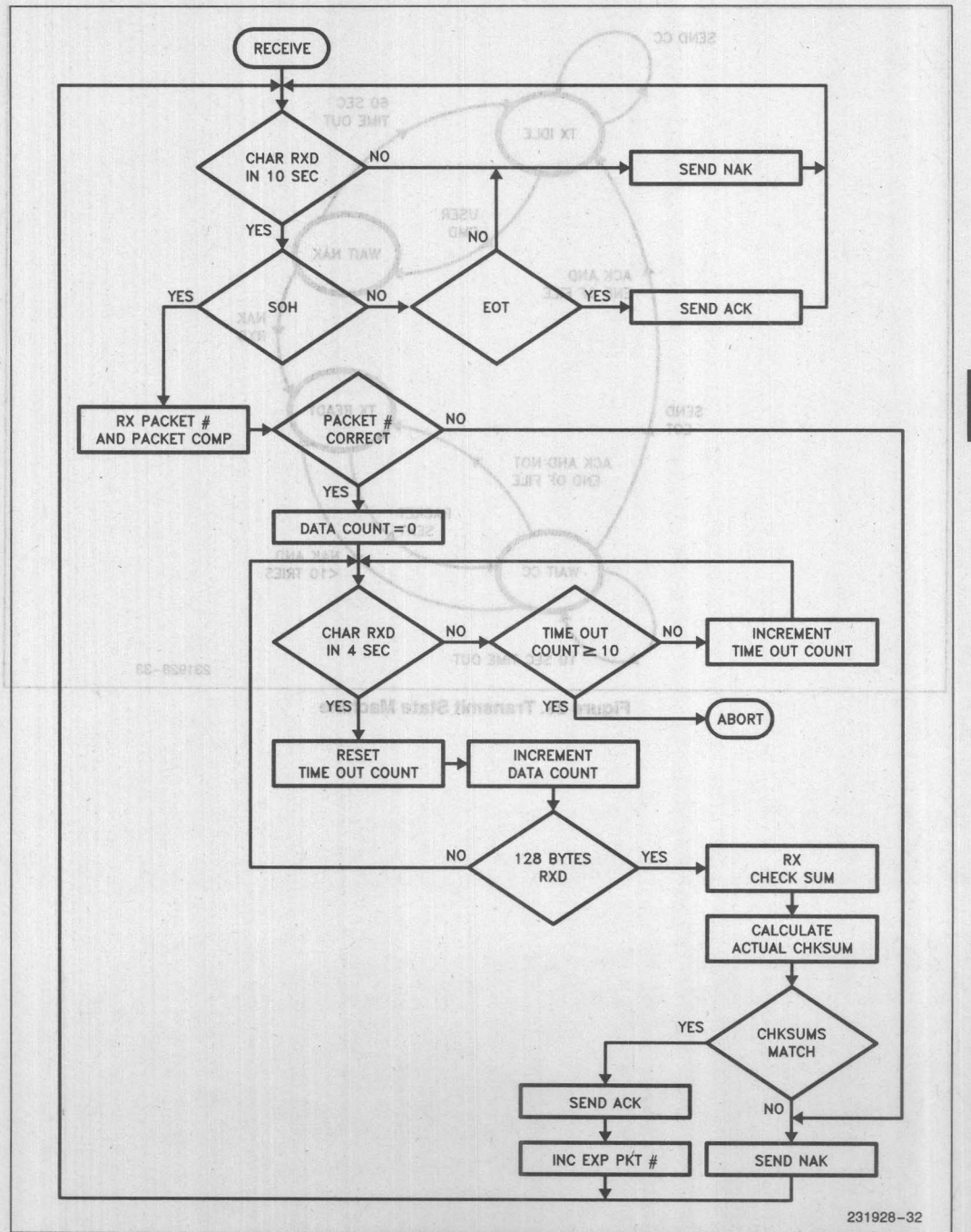


Figure 28. Protocol Flow for Receive Side of XMODEM

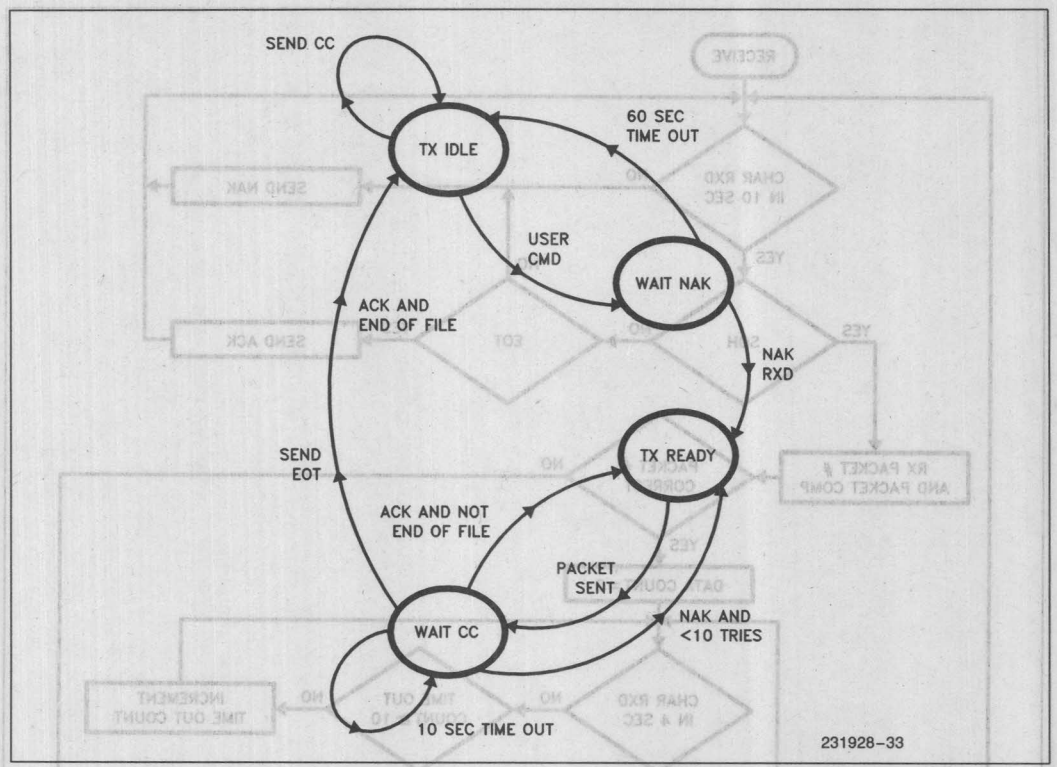


Figure 29. Transmit State Machine

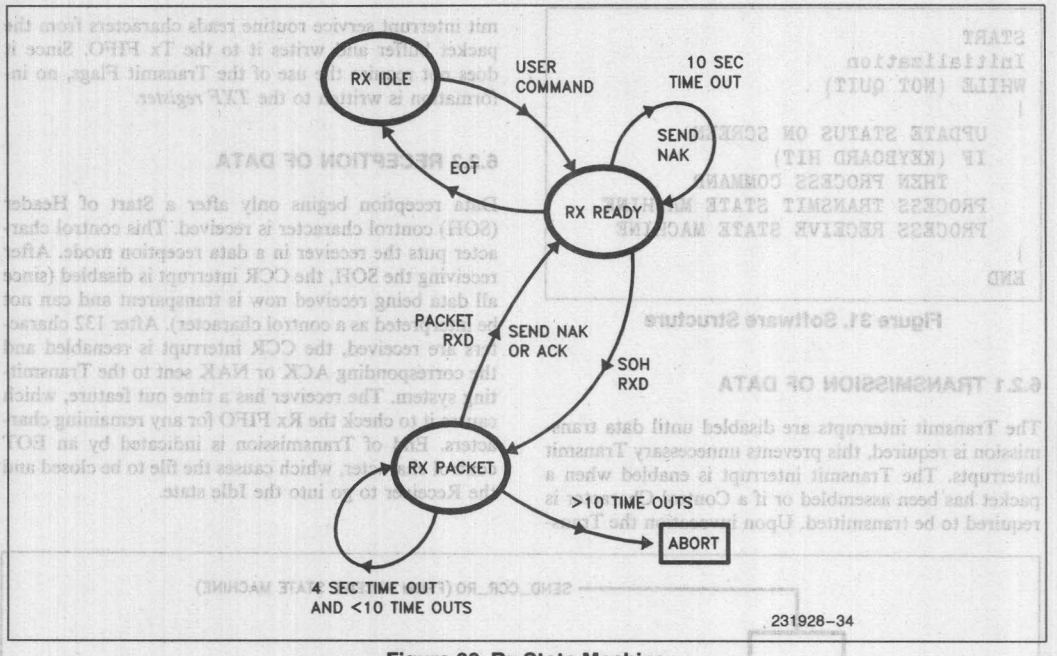


Figure 30. Rx State Machine

```
START
Initialization
WHILE (NOT QUIT)
{
   UPDATE STATUS ON SCREEN
   IF (KEYBOARD HIT)
      THEN PROCESS COMMAND
   PROCESS TRANSMIT STATE MACHINE
   PROCESS RECEIVE STATE MACHINE
}
END
```

Figure 31. Software Structure

6.2.1 TRANSMISSION OF DATA

The Transmit interrupts are disabled until data transmission is required, this prevents unnecessary Transmit interrupts. The Transmit interrupt is enabled when a packet has been assembled or if a Control Character is required to be transmitted. Upon invocation the Transmit interrupt service routine reads characters from the packet buffer and writes it to the Tx FIFO. Since it does not require the use of the Transmit Flags, no information is written to the *TXF register.*

6.2.2 RECEPTION OF DATA

Data reception begins only after a Start of Header (SOH) control character is received. This control character puts the receiver in a data reception mode. After receiving the SOH, the CCR interrupt is disabled (since all data being received now is transparent and can not be interpreted as a control character). After 132 characters are received, the CCR interrupt is reenabled and the corresponding ACK or NAK sent to the Transmitting system. The receiver has a time out feature, which causes it to check the Rx FIFO for any remaining characters. End of Transmission is indicated by an EOT control character, which causes the file to be closed and the Receiver to go into the Idle state.

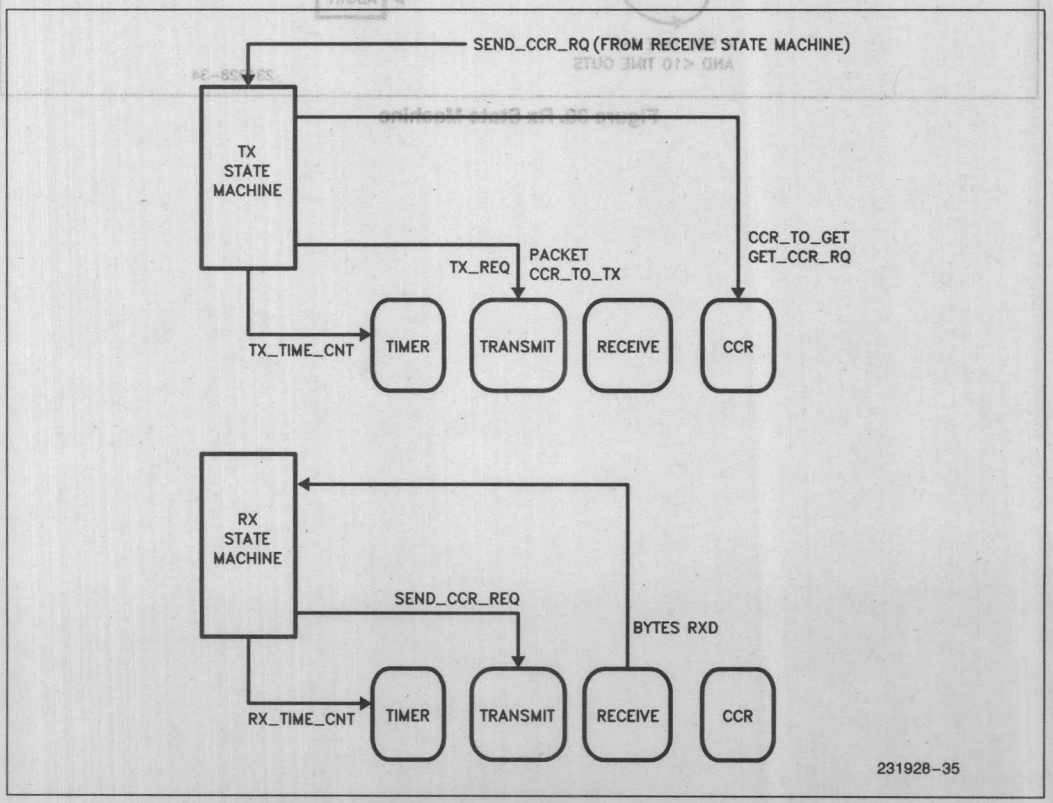


Figure 32. Using Flags for Communications with Interrupt Routine

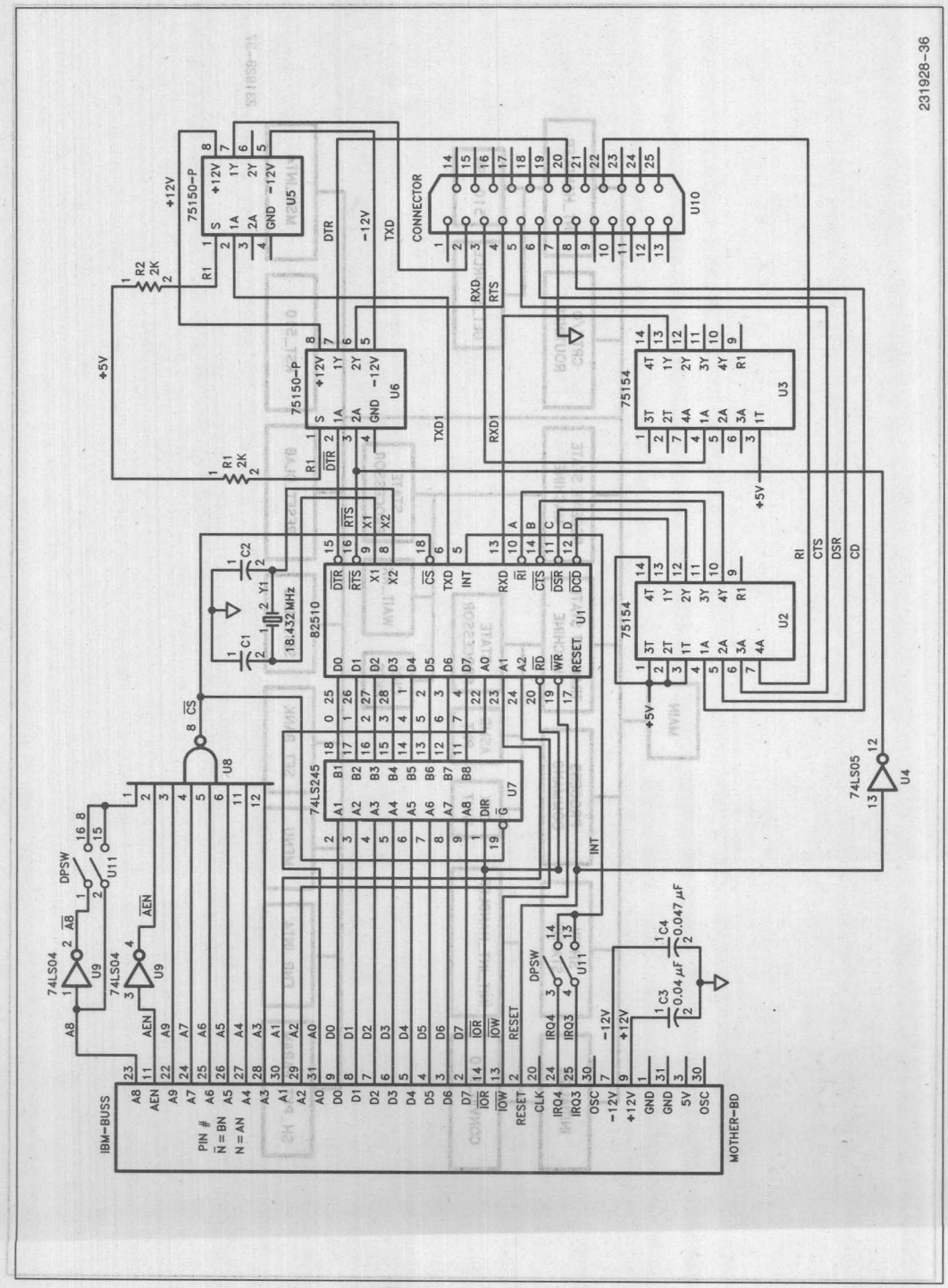


Figure 33. PC AT Adapter

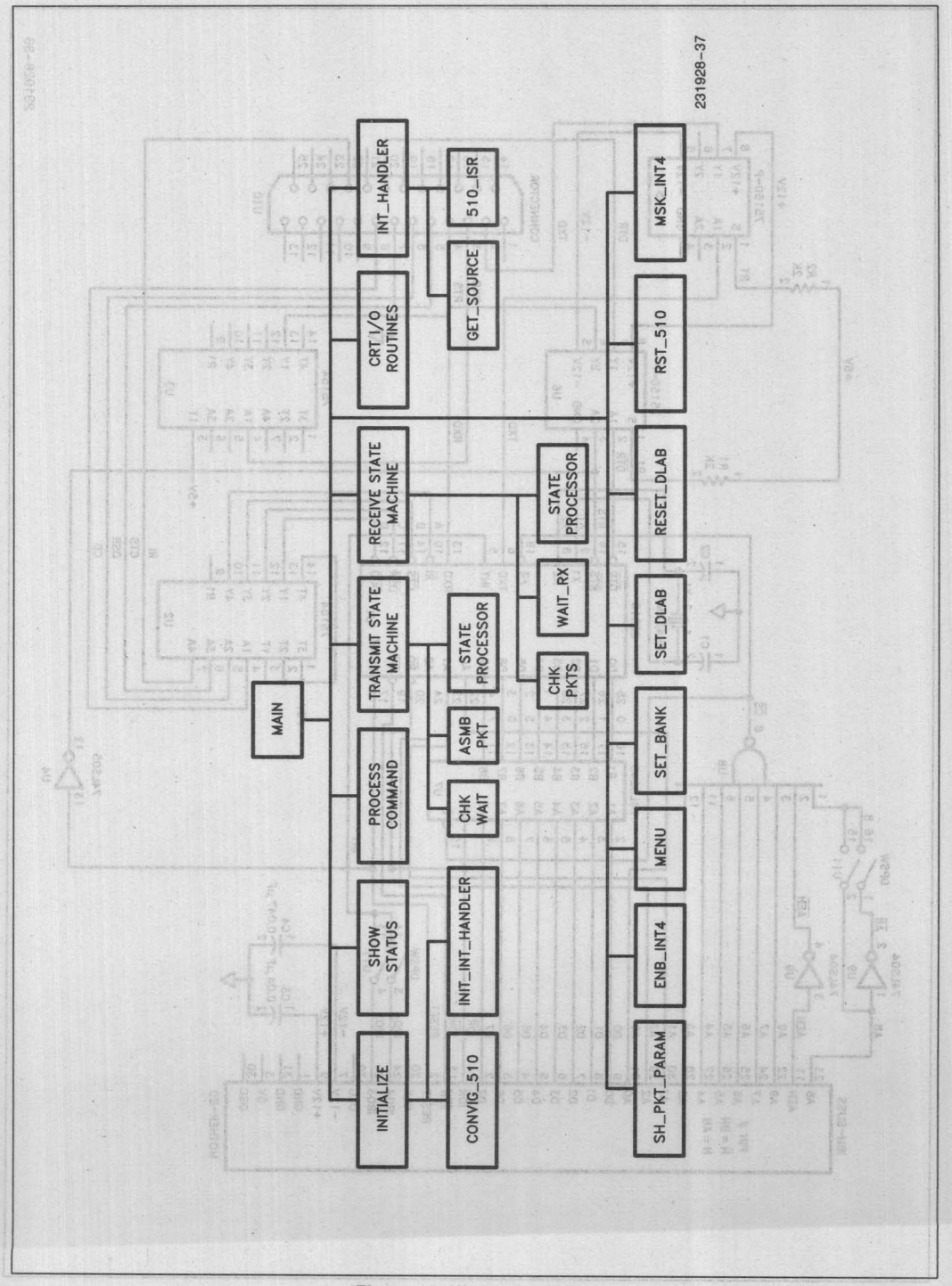


Figure 34. Hierarchy Chart

6.3 Software Listings

```
PAGE  1       MAIN PROGRAM   ftp.c  82510 XMODEM


  1. #include "C:\ftp\ftp.def"
  2. #include "C:\lc\fcntl.h"
  3. #include "C:\lc\stdlib.h"
  4. #include "C:\lc\stdio.h"
  5. /*************************************************/
  6. /***                                          *****/
  7. /***    SEPTEMBER 1986                        *****/
  8. /***                                          *****/
  9. /***    82510 XMODEM IMPLEMENTATION           *****/
 10. /*************************************************/
 11. int     eof =false;          /* end of file flag */
 12. int     rxpk =0;
 13. int     txrflg;
 14. int     rxrflg;
 15. int     exp_pkt_num = 1;     /* next packet number expected by receiver */
 16. int     pkst;
 17. int     rxtocnt;             /* Time Out counter for receiver */
 18. int     quit =false;
 19. int     key = 0;
 20. int     sohcnt =0;           /* # of SOH characters received */
 21. int     rxfcnt =0;           /* # of Rx FIFO Interrupts */
 22. int     ccrcnt =0;           /* # of Ctl-Char. Interrupts */
 23. int     tx_state =tx_idle;   /* Transmitter State Variable */
 24. int     rx_state = rx_idle;  /* Receiver State Variable */
 25. int     tx_cmd = inactive;   /* Indicates a Valid Tx Command was given */
 26. int     rx_cmd = inactive;   /* Indicates a valid Rx Command was issued */
 27.
 28. /* File to be Transmitted */
 29. char    tx_file_name[40] = "                                        ";
 30.
 31. /* File to be Received */
 32. char    rx_file_name[40] = "                                        ";
 33.
 34. int     send_ccr_req = inactive;     /* Flag - Request to Tx Ctl-Char */
 35. int     intvec =0;                   /* contains the  GIR vector */
 36. int     l;
 37. char    txdata [132];                /* Tx Buffer */
 38. char    rxbuf [128];                 /* Rx Buffer */
 39. char    rxdata [131];
 40. char    rx_f_buf [32000];            /* Rx File Stored in this buffer */
 41.
 42. /*****************************/
 43. /*** tx state variables *****/
 44. /*****************************/
 45. int     tx_indx;                     /* Pointer to the next character in the
 46.                                         buffer */
 47.
 48. struct  packet   {
 49.          char         head;
 50.          char         pack_num;
 51.          char         pack_cmpl;
 52.          char         buffer [128];
 53.          char         chksm;
 54.        } ;
 55.
 56. struct packet rxpack, txpack;
 57.
```

231928-38

82510 XMODEM Implementation

```
PAGE  2       MAIN PROGRAM   ftp.c  82510 XMODEM


 58.
 59. /*****************************************************/
 60. /****       tx State machine and interrupt       ******/
 61. /****       handler  flags                       ******/
 62. /*****************************************************/
 63.
 64. /** txm and tx fifo  **/
 65. int     tx_req =0;       /* Flag - indicates a request for transmission to
 66.                             82510 Interrupt Handler */
 67. int     ccr_to_tx = 0;  /* Actual Ctl-char to Transmit */
 68. int     tx_byte_cnt =0; /* Total # of Bytes Transmitted */
 69. int     pkts_sent =0;   /* # of Packets sent */
 70.
 71. /** Timer   **/
 72. int     tx_time_cnt =0; /* Transmitter Timer Counter */
 73.
 74. /** CCR **/
 75. int     get_ccr_rq =0;  /* Flag - Request to Receive Ctl-character */
 76. int     ccr_to_get  =0; /* Received Ctl-char value */
 77.
 78.
 79. /*************************************************************/
 80. /****                                              *****/
 81. /****       RX STATE VARIABLES                     *****/
 82. /****                                              *****/
 83. /****                                              *****/
 84. /*************************************************************/
 85.
 86. char    pk_chksm;        /* Calculated Chksum */
 87. int     eot_cnt =0;      /* # of EOTs Received */
 88. int     bad_pkt_cnt;     /* # of Bad Packets Received */
 89.
 90. /*****************************************************/
 91. /****       rx state machine and Interrupt       *****/
 92. /****       handler flags                        *****/
 93. /*****************************************************/
 94.
 95. /** rx fifo **/
 96. int     rx_byte_cnt =0;      /* # of Bytes Received */
 97.
 98. /** CCR **/
 99.
100. int     ctl_rxd_flg =0;      /* Flag-Indicating that a Ctl-Char. has been
101.                                receivd*/
102. int     rx_ctl_chr=0;        /* Actual Ctl-char received */
103.
104. /** Timer   **/
105. int     rx_time_cnt =0;      /* Receive Timer Count */
```

231928-39

82510 XMODEM Implementation (Continued)

```
PAGE  3       MAIN PROGRAM   ftp.c  82510 XMODEM


106.
107. /*****************************************/
108. /**          MAIN ROUTINE                ***/
109. /*****************************************/
110.
111. main ()
112. {
113. int      q,txfl,rxfl;
114. int      rval,v =0;
115. int      cmd = 0;
116. int      wn_status =0;
117. int      ecode = 0;
118. FILE     *fp;
119. FILE     *rxfp;
120. int      rwst;
121. int      wcst;
122. int      retx_cnt =0;            /* Retransmit count */
123. int      tocnt =0;               /* Time Out Count */
124. int      tx_secs,rx_secs;
125. int      i,s, lpcnt = 0;
126.
127.     CLR ();                     /* Clear Screen */
128.     MV_CURS (so_r,so_c);        /* Sign On Message */
129.     printf (s1);
130.     init ();                    /* Initialize 82510 and Variables */
131.     MENU ();                    /* Print Menu */
132.     enbint4 ();                 /* Enable Interrupts in 8259A */
133.     outp (ip00,eoi);            /* issue EOI    */
134.     outp ((bpa+3),0x22);        /* start timer B */
135.     lpcnt =0;                   /* Keeps Track of # of Loops */
136.
137. /*****************************************************/
138. /***            main while loop              ****/
139. /*****************************************************/
140.     while (quit==false)
141.     {
142.
143.     /*************************************/
144.     /** display protocol parameters     **/
145.     /*************************************/
146.
147.       ++ lpcnt;
148.       mv_curs (4,30);
149.       printf ("loop # = %u",lpcnt);
150.       mv_curs (4,50);
151.       printf ("rx int. cnt = %u",rxfcnt);
152.       mv_curs (5,50);
153.       printf ("ccr int cnt = %u",ccrcnt);
154.       mv_curs (4,1);
155.       printf ("interrupt vector = %u \n", intvec);
156.       q = inp (bpa+4);
157.       txfl = q & 0x07;
158.       mv_curs (5,1);
159.       printf ("TX FIFO = %u ",txfl);
160.       q = inp (bpa+4);
161.       rxfl = q & 0x70;
162.       mv_curs (6,1);
163.       printf ("RX FIFO = %u \n",rxfl/16);
164.       mv_curs (6,50);
165.       printf ("SOH count = %3u",sohcnt);
```

231928-40

82510 XMODEM Implementation (Continued)

```
PAGE  4       MAIN PROGRAM   ftp.c  82510 XMODEM

166.        mv_curs (7,1);
167.        printf ("bytes received %3u",rx_byte_cnt);
168.        mv_curs (7,30);
169.        printf ("Bytes Sent = %3u",tx_byte_cnt);
170.        mv_curs (7,50);
171.        printf  ("EOT count %3u", eot_cnt);
172.        mv_curs (5,30);
173.        printf ("pkts rxd = %3u",(exp_pkt_num-1));
174.        mv_curs (6,30);
175.        printf ("pkts sent = %3u", pkts_sent);
176.        tx_secs = tx_time_cnt/200;
177.        rx_secs = rx_time_cnt/200;
178.        open_wind (3,1,"Tx Timer");
179.        printf (" = %2u secs",tx_secs);
180.        open_wind (3,50,"Rx Timer");
181.        printf (" = %2u secs",rx_secs);
182.        mv_curs (8,1);
183.        printf ("Bad Packets Rxd = %3u",bad_pkt_cnt);
184.        mv_curs(8,30);
185.        printf ("# of ReTx packets = %3u",retx_cnt);
186.
187.  /* If Command Issued then process the Command */
188.        if ((key =kbhit()) ) 0)
189.           quit = process_cmd ();
190.
191.        else
192.        {
193.
194.          /*************************************************************/
195.          /*****          Process Tx STATE MACHINE                *****/
196.          /*****          revision 0                              *****/
197.          /*****                                                  *****/
198.          /*************************************************************/
199.
200.          switch (tx_state) {
201.              case tx_idle:
202.
203.          /*************************************************************/
204.          /****                                                    *****/
205.          /****  TRANSMITTER IDLE STATE                          *****/
206.          /****                                                    *****/
207.          /****              Checks for a Send Ctl-Char.           *****/
208.          /****              Checks for the Transmit Command       *****/
209.          /****                                                    *****/
210.          /****                                                    *****/
211.          /*************************************************************/
212.
213.
214.                  /* If Control Character to be Transmitted Then Transmit the
215.                     Control Character by setting the Tx_req Flag and enabling
216.                     the TxM and Tx FIFO interrupts */
217.
218.                  if ((send_ccr_req ==  active) && (!tx_req))
219.                  {
220.                      tx_req =ctl_chr;
221.                      tx_i_enb ();
222.                      while ( tx_req)0);
223.                      tx_i_dis ();
224.                      send_ccr_req=inactive;
225.                  }
```

231928-41

82510 XMODEM Implementation (Continued)

```
PAGE  5       MAIN PROGRAM   ftp.c  82510 XMODEM


226.                /* If the Transmit Command is issued then Wait for a NAK */
227.                if (tx_cmd ==  active)
228.                 {
229.                    tx_cmd = inactive;
230.                    get_ccr_rq =active;
231.                    tx_time_cnt =200*60; /* 60 sec. Time Out */
232.                    tx_state = wait_NAK;
233.                 }
234.                break;
235.
236.            case wait_NAK : /* Waiting for a NAK character to begin Tx */
237.
238.        /*********************************************************/
239.        /****  TRANSMITTER WAITING FOR A NAK TO BEGIN        *****/
240.        /****  TRANSMISSION.                                 *****/
241.        /****                                                *****/
242.        /****             Checks For Time Out                *****/
243.        /****                   or  NAK Received             *****/
244.        /*********************************************************/
245.
246.                wn_status = check_wait ();      /* Time Out or NAK Rcvd? */
247.                switch (wn_status)  {
248.
249.                    case time_out :             /* If Time Out then Abort
250.                                                   Transmission */
251.                        tx_state =tx_idle;
252.                        beep ();
253.                        prmsg ("Time OUT !!!! receiver not ready");
254.                        cll (tx_r, tx_c);
255.                        open_wind (tx_r,tx_c,"NONE");
256.                    break;
257.
258.                    case waiting :              /* if no Time Out and no NAK
259.                                                   rcvd then do nothing */
260.                    break;
261.
262.                    case rx_NAK :
263.                                                /* If NAK received then Open
264.                                                   file and advance to
265.                                                   Transmit Packet State */
266.
267.                        fp =fopen (tx_file_name,"rb" );
268.                        if (fp== NULL)
269.                        {
270.                         beep ();
271.                         prmsg ("ERROR !!! file does not exist");
272.                         cll (tx_r,tx_c);
273.                         open_wind (tx_r,tx_c, "none");
274.                         tx_state =tx_idle;
275.                        }
276.                     else
277.                            {
278.                            tx_state = tx_rdy;
279.                            txrflg = mkpkt;     /* First task for Tx
280.                                                   is to Prepare Packet */
281.                            wn_status = 0;      /* Reset Wait_NAK Flag */
282.                            }
283.                    break;
284.                    }
285.            break;/* end  wait nak */
```

231928-42

82510 XMODEM Implementation (Continued)

```
PAGE  6       MAIN PROGRAM   ftp.c  82510 XMODEM


286.            case    tx_rdy :
287.            /*********************************************************/
288.            /****  TANSMITTER READY TO TRANSMIT                  *****/
289.            /****               three stages of transmission     *****/
290.            /****                     prepare packet             *****/
291.            /****                     Int. Handler Transmitting  *****/
292.            /****                     or retransmit request      *****/
293.            /*********************************************************/
294.
295.                /* Any Control Character To Transmit? */
296.                if ((send_ccr_req == active) && (tx_req==0))
297.                   tx_req =ctl_chr;
298.
299.                /* Which Stage of transmission ?*/
300.                switch (txrflg)
301.                {
302.                case mkpkt:             /* Prepare Packet */
303.                    if  (tx_req==0)
304.                    {
305.                     asmbpkt (pkts_sent,fp);    /* Assemble Packet */
306.                     cpy2buf ();
307.                     tx_req =pkt;               /* Request Int. Handler
308.                                                   to Tx data in buffer */
309.                     txrflg =txmtg;             /* Start Transmission */
310.                     tx_indx =0;
311.                     tx_i_enb ();               /* Enable TxM and Tx FIFO
312.                                                   Interrupts */
313.
314.                    }
315.                break;
316.                case txmtg :
317.                    if (tx_req == 0)           /* Interrupt Handler Resets
318.                                                  this flag to 0, when 132
319.                                                  bytes are transmitted */
320.                    {
321.                      tx_indx =0;
322.                      prmsg ("packet transmitted");
323.                      get_ccr_rq =active;       /* Wait for ACK or NAK */
324.                      tx_time_cnt = 200*10;     /* 10 sec Time Out */
325.                      tx_state = wait_CC;       /* Wait for ctl Character */
326.                      txrflg =mkpkt;
327.                      tx_i_dis ();              /* Disable TxM and Tx FIFO
328.                                                   Interrupts */
329.                    }
330.                    else                       /* Tx_req not reset then
331.                       prmsg ("transmitting");  still transmitting */
332.                break;
333.                case retx :                    /* The Retransmit request is
334.                                                  issued by the Wait _CC
335.                                                  state */
336.                     outp((bpa+6),txen);       /* enable txm, flush tx fifo
337.                                                  & txm **/
338.                     tx_req = pkt;             /* transmit Packet,pkt. in
339.                                                  buffer */
340.                     txrflg =txmtg;            /* next task - ReTransmit */
341.                     tx_i_enb ();              /* Enable TxM and Tx FIFO
342.                                                  Interrupts */
343.                break;
344.                }
345.            break; /* End tx rdy case */
```

231928-43

82510 XMODEM Implementation (Continued)

```
PAGE  7       MAIN PROGRAM   ftp.c  82510 XMODEM

346.     case wait_CC :
347. /*****************************************************/
348. /****                                             *****/
349. /****  Transmitter State - Waiting For Ctl Char.   *****/
350. /****                                             *****/
351. /****            NAK - requests retransmission    *****/
352. /****            ACK - Transmit Next Packet       *****/
353. /****                                             *****/
354. /****                                             *****/
355. /*****************************************************/
356.
357.             wcst = check_wait ();          /* Check for one of the
358.                                               Following events:
359.                                                Time Out
360.                                                NAK Received
361.                                                ACK Received
362.                                               or Still Waiting */
363.             switch (wcst)
364.              {
365.               case time_out :
366.
367.                                             /* If Time Out, then restart
368.                                                Tx Timer. Abort if Time
369.                                                Out count is greater than
370.                                                ten */
371.                 if (tocnt )10)
372.                    {
373.                    wcst =0;
374.                    abort_tx ();
375.                    prmsg ("receiver not responding");
376.                    }
377.                 else
378.                    {
379.                     ++tocnt;                /* Inc. Time Out Count */
380.                     tx_time_cnt =200*10;
381.                    }
382.
383.               break;
384.
385.               case waiting :                /* if waiting, do nothing */
386.               break;
387.
388.               case rx_NAK :                 /* If NAK or Corrupted
389.                                                ctl-char. received */
390.               case rx_gen :
391.                 prmsg ("NAK received");
392.                 if (retx_cnt )10)           /* more than 10 attempts,
393.                                                then Abort*/
394.                   {
395.                    retx_cnt =0;
396.                    tocnt =0;
397.                    abort_tx ();
398.                    prmsg ("Bad link transmission aborted");
399.                   }
```

231928-44

82510 XMODEM Implementation (Continued)

2

```
PAGE  8       MAIN PROGRAM   ftp.c  82510 XMODEM


400.                else                            /* If Retransmit Count Not
401.                                                exceeded then go back to
402.                                                Transmit stage - task is
403.                                                retransmit */
404.                 {
405.                  txrflg =retx;
406.                  ++ retx_cnt ;
407.                  tx_state =tx_rdy;
408.                 }
409.               break;
410.
411.            case rx_ACK:       /* ACK Received*/
412.                prmsg ("ACK received");
413.                retx_cnt=0;
414.                tocnt = 0;
415.                ++pkts_sent;
416.                printf ("pkts_sent = %3u", pkts_sent);
417.                if (eof ==false)                /* If more data to transmit
418.                                                   then retrun to mkpkt
419.                                                   stage and tx new pkt. */
420.                  {
421.                   txrflg =mkpkt;
422.                   tx_state =tx_rdy;
423.                  }
424.                else
425.                 {
426.                  prmsg ("sending EOT");        /* if end of file , then
427.                                                   send EOT */
428.                  ccr_to_tx = EOT;
429.                  tx_req =ctl_chr;
430.                  tx_i_enb ();
431.                  while (tx_req != 0);          /* wait for Int. Handler
432.                                                   to reset flag */
433.                  tx_i_dis ();
434.                  get_ccr_rq =active;           /* wait for Ack */
435.                  while (get_ccr_rq ==active);
436.                  prmsg ("EOT acknowledgement received");
437.                  if (ccr_to_get == ACK)        /* ACK rxd , Close File */
438.                  {
439.                     s = fclose (fp);
440.                     abort_tx();
441.                     prmsg ("file transmission complete");
442.                  }
443.                  tx_state =tx_idle;            /* Return to Idle */
444.                 }
445.            break;
446.            } /* /end wait_cc case */
447.         break;
448.         } /* end switch tx state */
```

231928-45

82510 XMODEM Implementation (Continued)

```
PAGE  9       MAIN PROGRAM   ftp.c  82510 XMODEM


/*****************************************************/
/*****          Process Rx STATE MACHINE          *****/
/*****          revision 0                        *****/
/*****                                            *****/
/*****************************************************/
        switch (rx_state)
        {
            case rx_idle:
            /*************************************************/
            /****                                         *****/
            /**** RECEIVER IDLE:                         *****/
            /****                                         *****/
            /****              waits for user command     *****/
            /****              before sending NAKs        *****/
            /****                                         *****/
            /****                                         *****/
            /*************************************************/

                if (rx_cmd == active)       /* If receive Command is issued
                                               then start Rx timer and change
                                               Receiver state to ready */
                    {
                        rx_state = rx_rdy;
                        rx_time_cnt =200*10;
                        rx_cmd = inactive;
                    }
            break;

            case rx_rdy:
            /*************************************************/
            /****                                         *****/
            /**** RECEIVER READY:                         *****/
            /****              sends NAK upon Time Out    *****/
            /****              or checks for SOH          *****/
            /****              or EOT ctl-char.           *****/
            /****                                         *****/
            /****                                         *****/
            /*************************************************/

                rxrflg =wait_rx ();     /* Checks Rx Timer and returns -
                                           Time Out if expired
                                           waiting if not expired
                                           SOH if SOH ccr received
                                           EOT if EOT ccr received */
                switch (rxrflg)
                {
                 case waiting :         /* If waiting then do nothing */
                 break;

                 case  SOH :            /* If SOH received, then go into
                                           data reception mode and change Rx
                                           Timer count to 4 secs */
                    ++ sohcnt;
                    rx_state =rx_pkt;
                    rx_time_cnt =200*4; /* four second time out */
                    rxtocnt =0;
                break;

                                                              231928-46
```

82510 XMODEM Implementation (Continued)

```
PAGE 10      MAIN PROGRAM   ftp.c  82510 XMODEM

507.                case time_out:          /* if time out & not in the midst of
508.                                           packet reception then send NAK */
509.                    if (( exp_pkt_num ==1) && (rx_byte_cnt ==0))
510.                    {
511.                        prmsg ("rx time out !!!!! sending NAK");
512.                        if (send_ccr_req ==inactive)
513.                        {
514.                            ccr_to_tx =NAK;
515.                            send_ccr_req =active;
516.                        }
517.                    }
518.                    rx_time_cnt =200*10;
519.                break;
520.
521.                case EOT:                   /* If End Of Text rcvd,
522.                                               and data rcvd then
523.                                               send ACK and save all
524.                                               packets received in
525.                                               file */
526.                    ++ eot_cnt;
527.                    open_wind (22,50,"End of Text");
528.                    if (exp_pkt_num )1)
529.                    {
530.                      if (send_ccr_req == inactive) /* Send ACK */
531.                      { send_ccr_req =active;
532.                        ccr_to_tx =ACK;
533.                      }
534.                      rx_state =rx_idle;            /* Receiver Returns to
535.                                                       Idle */
536.                                                    /* create file */
537.                      rxfp =fopen (rx_file_name,"ab+");
538.                      rwst =fwrite (&rx_f_buf[0],128,exp_pkt_num-1,rxfp);
539.                      if (rwst (1)
540.                            {
541.                             prmsg ("Write file error ");
542.                             printf ("error = %4u",(rwst=ferror(rxfp)));
543.                            }
544.                      rval =fclose (rxfp);
545.                      if (rval ==0)
546.                          prmsg ("file received");
547.                      else
548.                        prmsg ("Error in closing file ");
549.
550.                    }
551.                break;
552.                }
553.            break;
554.
555.    /*.PA*/
556.            case rx_pkt: /* Packet reception */
557.            /*************************************************/
558.            /****                                         *****/
559.            /****   RECEIVE PACKET STATE                  *****/
560.            /****                                         *****/
561.            /****              checks for Time Out        *****/
562.            /****              or 131 bytes received      *****/
563.            /****              which signals the end of packet *****/
564.            /****                                         *****/
565.            /*************************************************/
566.
```

231928-47

82510 XMODEM Implementation (Continued)

```
PAGE 11       MAIN PROGRAM   ftp.c  82510 XMODEM


567.            /* If valid Rx Time Out , i.e. no data received for 4 secs then
568.               check Rx FIFO for characters and read if any available */
569.            if ((rx_time_cnt ==0) && (rx_byte_cnt (131))
570.            {
571.                rxfl = ((inp (bpa +4) & 0x70)/ 0x10);          /* check Rx FIFO
572.                                                                  Level */
573.                if ((rxtocnt )= 10) && (rxfl (=0))           /* if more than
574.                                                                  10 attempts
575.                                                                  and no data
576.                                                                  then abort
577.                                                                  transmit */
578.                {
579.                    rx_state =rx_idle;
580.                    prmsg (" Receiver Time Out, no DATA");
581.                    rxtocnt =0;
582.                }
583.
584.                else
585.                /* otherwise restart Rx Timer, and read data from 510 */
586.                {
587.                 if (rxfl != 0)                        /* Rx FIFO level ) 0 */
588.                 {
589.                     rx_time_cnt =200*5;
590.                     rxfl = ((inp ( bpa +4) & 0x70)/0x10);
591.                     while ( rxfl != 0)                /* Read from FIFO */
592.                     {
593.                         rxdata [rx_byte_cnt] = inp (bpa);
594.                         ++ rx_byte_cnt;
595.                         ++ rxfcnt;
596.                         -- rxfl;
597.                     }
598.                     rxtocnt = 0;
599.                 }
600.                 else
601.                 {
602.                     ++ rxtocnt;                       /* inc. receive TimeOut
603.                     rx_time_cnt =200*4;                  Count */
604.                 }
605.                }
606.            }
607.            else
608.            {
609.                if (rx_byte_cnt == 131)               /* Packet Received */
610.                {
611.                    rx_byte_cnt =0;
612.                    rxtocnt =0;
613.                    pkst =chkpkt (exp_pkt_num);       /* Check Packet */
614.                                                      /* returns EOK if Packet
615.                                                         without errors */
616.        /*.PA*/
617.                    if ((pkst == eok) || (pkst ==eold))
618.                    {
619.                        prmsg ("sending ACK");
620.                        for (i=0; i(128; i++)
621.                            rxbuf [i] = rxdata [i+2];
622.                        /** write packet to buffer  **/
623.                        if (pkst ==eok)
624.                        {
625.                                                      /* copy to main file
626.                                                         buffer */
```

231928-48

82510 XMODEM Implementation (Continued)

```
PAGE 12       MAIN PROGRAM   ftp.c  82510 XMODEM


627.                            buf_cpy (exp_pkt_num);
628.                            ++exp_pkt_num;
629.                        }
630.                        else
631.                            prmsg ("old packet retransmitted");
632.                    }
633.
634.                    else
635.                        sh_pkt_param ();            /* If error then show
636.                                                       packet #, chksum and
637.                                                       packet complement */
638.                    rx_state = rx_rdy;
639.                    mskint4 ();                     /* Enable Ctl-Chr int*/
640.                    set_bank (00);
641.                    outp ((bpa+1),(inp(bpa+1)|ccien));
642.                    set_bank (01);
643.                    enbint4 ();
644.                    send_ccr_req =active;           /* Send ACK */
645.                    ccr_to_tx =ACK;
646.
647.                }
648.
649.            }
650.            break;
651.
652.        } /** end switch rx state **/
653.
654.
655.    } /* end else */
656.
657. } /* end while quit */
658.
659. rst510 ();                                 /* reset 82510 */
660. outp ((bpa + 1),00);                       /* disable 82510 interrupts */
661.
662.
663. cmd = 0x10;                                /* disable 8259A interrupt */
664. v=inp (0x21);                              /*       00010000          */
665. cmd = (v | cmd);
666. outp  (0x21,cmd);
667. clr ();
668. ecode  = 0;
669. _exit (ecode);
670.
671. }/* end main */
672.
```

231928-49

82510 XMODEM Implementation (Continued)

```
PAGE 13       MAIN PROGRAM   ftp.c  82510 XMODEM


673.
674. rst510 ()
675. /*****************************************************/
676. /****                                             *****/
677. /****    RESET 82510 to default wake up mode      *****/
678. /****                                             *****/
679. /****                                             *****/
680. /****                                             *****/
681. /*****************************************************/
682.
683. {
684. set_bank (01);
685. outp ((bpa+7),0x10);
686. }
687.
688.
689. menu ()
690. /*************************************************/
691. /**     displays the menu on the                **/
692. /**     screen.                                 **/
693. /**                                             **/
694. /**                                             **/
695. /**                                             **/
696. /*************************************************/
697. {
698.
699.
700.     open_wind (1,1,"baud rate");
701.     printf (" = 1200  ");
702.     open_wind (1,22,"char. size");
703.     printf (" = 8 bits");
704.     open_wind (1,45 , "Parity");
705.     printf (" disabled");
706.     open_wind (1,68, "Stop Bits");
707.     printf (" = 2");
708.     mv_curs (2,1);
709.     printf ("user messages :");
710.     mv_curs (10,15);
711.     printf ("(1) TRANSMIT FILE : ");
712.     OPEN_WIND (tx_r,tx_c,"none");
713.     mv_curs (12,15);
714.     printf ("(2) RECEIVE FILE : ");
715.     OPEN_WIND (rx_r,rx_c,"none");
716. }
717.
```

231928-50

82510 XMODEM Implementation (Continued)

```
PAGE 14       MAIN PROGRAM   ftp.c  82510 XMODEM


718.
719. init    ()
720. /*************************************************/
721. /**  Intialixes Software and Configures          **/
722. /**  the 82510. Also sets up the interrupt       **/
723. /**   Handler.                                   **/
724. /**                                              **/
725. /**                                              **/
726. /*************************************************/
727.
728. {
729. tx_time_cnt =200;
730. rx_time_cnt =2000;
731. initpack ();
732. clms ();
733. init_ih ();                    /* Set up interrupt handler */
734. config_510 ();                 /* Configure 82510 */
735. set_bank (01);                 /* Switch to Bank one for operation */
736.
737. }
738. initpack ()
739. /*************************************************/
740. /**                                              **/
741. /**     Intialixes Tx Buffer to NULs             **/
742. /**                                              **/
743. /*************************************************/
744.
745. {
746. int     i;
747.
748. txpack.head = SOH;
749. rxpack.head =SOH;
750. txpack.pack_num =0;
751. rxpack.pack_num =0;
752. txpack.pack_cmpl = 0;
753. rxpack.pack_cmpl = 0;
754. for (i=0; i <129; i++)
755.     {
756.         rxpack.buffer[i] =NUL;
757.         txpack.buffer[i] =NUL;
758.     }
759. txpack.chksm =0;
760. rxpack.chksm =0;
761. }
762.
763. enbint4 ()
764. /*************************************************/
765. /**                                              **/
766. /**     Enables INT4  in the 8259A               **/
767. /**                                              **/
768. /*************************************************/
769.
770. {
771. int     int_enb = 0xEF;
772. int     v;
773.
774. v=inp (ip01);              /* 11101111 */
775. int_enb = (v & int_enb);
776. outp  (ip01,int_enb);
777. }
```

231928-51

82510 XMODEM Implementation (Continued)

```
config_510 ()
/***********************************************/
/**                                           **/
/**      Configure the 82510                  **/
/**                                           **/
/***********************************************/
{
int val;

set_bank (02);              /*****************************************/
val = 0x00;                 /* IMD - Rx FIFO depth =4, auto ack,normal  */
outp ((bpa + 4), val);      /* local loopback                           */
val =0x78   ;               /* RMD - ASCII CCR,disable dpll,7/16 sampl  */
outp ((bpa +7),val);        /* window,absolute start bit sampling       */
val =0x00       ;           /* TMD - manual  mode, 2 stop bits          */
outp ((bpa +3),val);        /* no 9-bit char, no s/w parity             */
val =0x30;                  /* FMD - Rx fifo Threshold = 3              */
outp ((bpa+1),val);         /* Tx fifo threshold =0                     */
val =0x80;                  /* RIE - Enable rx interrupts               */
outp ((bpa+6),val);         /*                                          */
set_bank (03);              /* MODEM CONFIGURATION                      */
val = 0x50;                 /* CLCF - 16X, BRGA                         */
outp ((bpa),val);           /*                                          */
val =0xd8;                  /* BBL  - for 5ms base                      */
set_dlab (03);              /*                                          */
outp ((bpa), val);          /*                                          */
val =0xb4;                  /*                                          */
outp ((bpa+1),val);         /* BBH - for 5 ms base                      */
reset_dlab (03);            /*                                          */
val = 0x00;                 /* BBCF - sys clk source, timer mode        */
outp ((bpa+3),val);         /*                                          */
val =0x02;                  /* TMIE - Timer B interrupt enable          */
outp ((bpa+6),val);         /*                                          */
set_bank (00);              /* BANK 0 FOR GENERAL CONFIG                */
val =blkenb;                /* GER -  enable timer, rx, CCR             */
outp ((bpa+1),val);         /* block interrupts                         */
val = 0x07;                 /* LCR - disable parity, 8 bit char         */
outp ((bpa+3),val);         /*                                          */
set_dlab (00);              /*                                          */
val = 0xE0;                 /* BRGA divisor =01E0H for 1200             */
outp (bpa,val);             /*****************************************/
val = 0x01;
outp ((bpa+1),val);
reset_dlab (00);
}
```

231928-52

82510 XMODEM Implementation (Continued)

```
PAGE 16       MAIN PROGRAM   ftp.c  82510 XMODEM


838. set_dlab (bank)
839. /*************************************************/
840. /**                                             **/
841. /**     Set DLAB bit to allow access to         **/
842. /**     Divisor Registers                       **/
843. /**                                             **/
844. /*************************************************/
845.
846. int bank;
847. {
848. int     inval;
849. set_bank (00);
850. inval = inp(bpa +3);
851. inval =inval ! 0x80;          /* set dlab in LCR*/
852. outp ((bpa+3),inval);
853. set_bank (bank);
854. }
855.
856. reset_dlab (bank)
857. /*************************************************/
858. /**                                             **/
859. /**     Reset DLAB bit of LCR                   **/
860. /**                                             **/
861. /*************************************************/
862.
863. int bank;
864. {
865. int     inval;
866. set_bank (00);
867. inval = inp(bpa +3);
868. inval = (inval & 0x7f);      /* dlab = 0 in LCR*/
869. outp ((bpa+3),inval);
870. set_bank (bank);
871. }
```

231928-53

82510 XMODEM Implementation (Continued)

```
PAGE 17       MAIN PROGRAM   ftp.c  82510 XMODEM

872. /*********************************************************/
873. /****                                                 *****/
874. /****    82510 interrupt service routine              *****/
875. /****                                                 *****/
876. /****    82510 Interrupt sources:                     *****/
877. /****                          TxM    TX FIFO         *****/
878. /****                          CCR    RX FIFO         *****/
879. /****                                 TIMER B         *****/
880. /****                                                 *****/
881. /****   Identifies and services the 82510 interrupt   *****/
882. /****   source requesting service.                    *****/
883. /****                                                 *****/
884. /****                                                 *****/
885. /*********************************************************/
886.
887. isr_510 ()
888. {
889. int     source ;
890. int     cmd_b;
891. int     st_b;
892. int     i;
893. int     ctlc;
894. int     flgs;
895. int     girval;                          /* Stores Temp. Value of GIR */
896. int     rxflvl;
897. int     tx_char;
898.
899. girval =inp (bpa+2);                     /* Save Bank register in temp.
900.                                             location */
901. outp ((bpa+2),0x20)
902. source = getsrc ();                      /* Get Vector From GIR 123 */
903. intvec =source;
904. switch (source)  {                      /* Service the Source */
905.
906.     case timer :
907. /*********************************************************/
908. /****                                             *****/
909. /****   TIMER SERVICE ROUTINE                     *****/
910. /****                    decrements tx counter    *****/
911. /****                    decrements rx counter    *****/
912. /****                                             *****/
913. /*********************************************************/
914.
915.         st_b = inp (bpa+3);              /* Decrement Transmit Counter */
916.         if (tx_time_cnt )0)
917.             tx_time_cnt =tx_time_cnt - 1;
918.         if (rx_time_cnt )0)              /* Decrement Receive Counter */
919.             rx_time_cnt =rx_time_cnt - 1;
920.         cmd_b = 0x22;
921.         outp ( (bpa+3),cmd_b );          /* restart timer */
922.         outp ((bpa+7),0x08);             /* manual ack */
923.         break;
```

231928-54

82510 XMODEM Implementation (Continued)

```
PAGE 18       MAIN PROGRAM   ftp.c  82510 XMODEM

924.    case txm   :
925.    case txf   :
926.    /*********************************************************/
927.    /****                                                 *****/
928.    /****  TRANSMITTER SERVICE ROUTINE                    *****/
929.    /****                                                 *****/
930.    /****              transmits Four characters          *****/
931.    /****              and resets tx_req flag when        *****/
932.    /****              whole packet transmitted           *****/
933.    /****                                                 *****/
934.    /*********************************************************/
935.
936.            if (tx_req )0)                  /* If data to send */
937.            {
938.                if (tx_req == pkt)          /* request to send Packet */
939.                {
940.                    for (i =0 ; i(4 ;i++)
941.                    {
942.                        tx_char = txdata [i + tx_indx];
943.                        outp (bpa,tx_char);
944.                    }
945.
946.                    tx_indx +=4;
947.                    tx_byte_cnt +=4;
948.
949.                    if (tx_indx )=132)      /* if 132 char. sent then */
950.                        tx_req = 0;         /* reset Tx request */
951.
952.                }
953.                else
954.                {
955.                    if (tx_req ==ctl_chr)   /* if ctl char. transmission
956.                                               requested, then transmit the
957.                                               char. in ccr_to_tx */
958.                         outp (bpa, ccr_to_tx);
959.                    tx_req =0;
960.                }
961.            }
962.            else
963.                {                           /* if no data to transmit */
964.                                            /* then disable tx interrupts */
965.                    set_bank (00);
966.                    outp ((bpa+1), (inp(bpa) &txidb));
967.                    set_bank (01);
968.                }
969.        outp ((bpa+7),0x08);                /* issue manual acknowledge */
970.    break;
971.
```

231928-55

82510 XMODEM Implementation (Continued)

```
PAGE 19       MAIN PROGRAM   ftp.c  82510 XMODEM


972.    case ccr   :
973.     /*************************************************************/
974.     /****                                                 *****/
975.     /****    Control Character Service Routine            *****/
976.     /****                                                 *****/
977.     /****              if control char = NAK or ACK       *****/
978.     /****                 inform    transmitter           *****/
979.     /****              if SOH  or EOT                     *****/
980.     /****                 inform receiver                 *****/
981.     /*************************************************************/
982.
983.                 ++ccrcnt;
984.                 flgs =inp (bpa +5);          /* read RST register to service
985.                                                 RxM interrupt */
986.                 flgs =inp (bpa+1);
987.                 ctlc =inp (bpa);
988.                 if ((flgs & 0xFF) ==0x48)  /* if no errors and ctl. char */
989.                 {                          /* then process control char. */
990.                                            /* and send to tx or rx state */
991.                 switch (ctlc)
992.                  {
993.                     case NAK:
994.                     case ACK:
995.                         if (get_ccr_rq == active)
996.                             {               /* inform transmitter that
997.                                                ctl. char. received */
998.                               get_ccr_rq =inactive;
999.                               ccr_to_get =ctlc;
1000.                             }
1001.
1002.                     break;
1003.
1004.                     case SOH:
1005.                     case EOT:
1006.                         if (ctlc ==SOH)     /* if SOH disable CCR int. */
1007.                         {
1008.                           set_bank (00);
1009.                           outp ((bpa+1),(inp(bpa+1)& ccidb));
1010.                           set_bank (01);
1011.                         }
1012.                         if (rx_state == rx_rdy)  /* if receiver waiting for
1013.                                                     SOH and ready to rcv
1014.                                                     then inform receiver of
1015.                                                     a valid ctl. char. */
1016.                          {
1017.                             ctl_rxd_flg =active;
1018.                             rx_ctl_chr =ctlc;
1019.                          }
1020.
1021.                     break;
1022.
1023.
1024.                  }
1025.                 }
1026.                 outp ((bpa+7),0x08);          /* issue manual ack. */
1027.    break;
```

231928-56

82510 XMODEM Implementation (Continued)

```
                    rx_time_cnt =200*5;      /* reset Rx Timer  to indicate
                                                char. received before time out */

                    rxflvl = ((inp ( bpa +4) & 0x70)/0x10);
                    while ( rxflvl != 0)     /* Check Rx FIFO level and read
                                                data if FIFO not empty */
                    {
                        rxdata [rx_byte_cnt] = inp (bpa);
                        ++ rx_byte_cnt;
                        ++ rxfcnt;
                        -- rxflvl;
                    }
                    outp ((bpa+7),0x08);     /* issue manual acknowledge */
        break;
        default   :
                                             /* if invalid source then issue a
                                                manual acknowledge */
                    outp ((bpa+7),0x08);
        break;
        }
outp ((bpa+1),girval);                       /* Restore Original value of Bank
                                                register to return the 82510 to
                                                original Bank  */

outp (ip00,eoi);                             /* issue end of int. to 8259A*/
}

Set_bank (bank_num)
int     bank_num;
/*************************************************************/
/****    PROCEDURE SET_BANK                                   *****/
/****    switches 82510 register bank to                      *****/
/****    given value.                                         *****/
/*************************************************************/
{
int     port;
int     bank_reg_val;

bank_reg_val    =bank_num * 0x20;
port = gir_addr +bpa;
outp (port, bank_reg_val);          /* output value to bank register */
}
```

231928-57

82510 XMODEM Implementation (Continued)

```
PAGE 21       MAIN PROGRAM   ftp.c  82510 XMODEM


1086. getsrc ()
1087. /*************************************************/
1088. /**      read GIR and returns the                **/
1089. /**      source Vector                           **/
1090. /**                                              **/
1091. /**      Timer       - 05 Hex                    **/
1092. /**      Tx Machine  - 04 Hex                    **/
1093. /**      CCR         - 03 Hex                    **/
1094. /**      Rx FIFO     - 02 Hex                    **/
1095. /**      Tx FIFO     - 01 Hex                    **/
1096. /**                                              **/
1097. /*************************************************/
1098.
1099. {
1100. int     v,src;
1101.
1102. v=inp (bpa +2);                    /* read GIR */
1103. src = v & 0x0E;                    /* Mask out all bits except for
1104.                                       bits 1,2 and 3 */
1105. src = src/2;
1106. return(src);
1107. }
1108.
1109. process_cmd ()
1110. /*********************************************************/
1111. /****                                                 ****/
1112. /**** PROCESS COMMAND                                 ****/
1113. /****                Processes User commands          ****/
1114. /****                1 - Transmit                     ****/
1115. /****                2 - Receive                      ****/
1116. /****                * - Reset 82510                  ****/
1117. /****                0 - quit                         ****/
1118. /****                r - Reinitialize 82510           ****/
1119. /****                ! - system monitor               ****/
1120. /****                                                 ****/
1121. /****                                                 ****/
1122. /*********************************************************/
1123.
1124.
1125. {
1126. int     r;
1127. int     exflg =false ;
1128. int     excp ;
1129.
1130.     r = getch ();
1131.     switch (r)  {
1132.
1133.         case '0' :
1134.                 exflg = true;              /* exit */
1135.                 break ;
```

231928-58

82510 XMODEM Implementation (Continued)

```
PAGE 22       MAIN PROGRAM   ftp.c  82510 XMODEM


1136.              case '1' :
1137.                  if (tx_state == tx_idle)        /* Transmit Command only
1138.                                                     accepted if idle */
1139.                  {
1140.                      CLMS ();
1141.                      CLL (tx_r,tx_c);
1142.                      MV_CURS (tx_r,tx_c);
1143.                      printf ("file :");          /* Get name of file to Tx */
1144.                      scanf ("%s", &tx_file_name);
1145.                      cll (tx_r,tx_c);
1146.                      open_wind (tx_r,tx_c,"transmitting");
1147.                      open_wind (tx_r,tx_c+14,tx_file_name);
1148.                      tx_cmd = active;            /* Activates flag to signal
1149.                                                     Transmit idle state */
1150.                  }
1151.                  else
1152.                  {
1153.                   beep ();
1154.                   prmsg ("transmission in progress");
1155.                  }
1156.          break;
1157.          case '2' :
1158.                  CLMS ();
1159.                  CLL (rx_r,rx_c);
1160.                  MV_CURS (rx_r,rx_c);
1161.                  printf ("file :");              /* Get rx file name */
1162.                  scanf ("%s", &rx_file_name);
1163.                  cll (rx_r,rx_c);
1164.                  open_wind (rx_r,rx_c,"enabled");
1165.                  open_wind (rx_r,rx_c+14,rx_file_name);
1166.                  rx_cmd =active;                 /* Activate flag to signal
1167.                                                     rx state machine */
1168.                  break;
1169.          case '*' :
1170.                  rst510 ();                      /* reset 82510 */
1171.                  open_wind (24,30,"device reset");
1172.                  break;
1173.          case 'r' :
1174.                  rst510 ();
1175.                  init ();                        /* reinitialize 82510 */
1176.                  enbint4 ();
1177.                  beep ();
1178.                  prmsg (" 82510 reinitialized");
1179.                  break;
1180.          case '!' :
1181.              excp = system ("d:\micom");
1182.          default:
1183.              BEEP ();
1184.              prmsg ("incorrect command, reenter");
1185.              break;
1186.      }
1187.      if (exflg == true)                          /* if exit command issued,
1188.                                                     then quit program */
1189.          return (true);
1190.      else  return (false);
1191.  )  /* end of command processing */
```

231928-59

82510 XMODEM Implementation (Continued)

```
PAGE 23       MAIN PROGRAM   ftp.c  82510 XMODEM


1192. asmbpkt (pkts_sent,fp)
1193. /*************************************************************/
1194. /**     Reads file to be transmitted and puts               **/
1195. /**     the data into the proper xmodem packet format       **/
1196. /*************************************************************/
1197.
1198. int     pkts_sent;                          /* this value is used to
1199.                                                get the next pkt # */
1200. FILE    *fp;
1201. {
1202. int     sum =0;
1203. int     i,blkcnt;
1204. int     st,ft;
1205. char    cpkt,cpkcmp;
1206.
1207.  blkcnt =fread (&txpack.buffer[0],128,1, fp);   /* read 128 bytes */
1208.  if (blkcnt (1)
1209.    {
1210.      if ((st=feof(fp)) )0 && !(ft=ferror(fp)))
1211.       {
1212.         eof = true;                          /* if end of file then
1213.                                                 signal EOF */
1214.         beep ();
1215.         prmsg ("EOF !!!!!!!!!!!!");
1216.       }
1217.          else
1218.         if (ft )0)
1219.            {
1220.             beep ();
1221.             prmsg ("READ ERROR !!!!!!!!!!!!");
1222.             tx_state=tx_idle;
1223.            }
1224.    }
1225.
1226.  cpkt =pkts_sent +1;
1227.  txpack.pack_num = cpkt;
1228.  cpkcmp =~txpack.pack_num;
1229.  txpack.pack_cmpl = cpkcmp;                  /* one's complement of
1230.                                                 packet number */
1231.  for (i=0; i (128; i++)
1232.      sum = sum+txpack.buffer[i];
1233.  txpack.chksm = sum % 255;                   /* checksum calculated */
1234.
1235. }
1236.
1237.
1238. cpy2buf ()
1239. /*******************************/
1240. /** copy packet to tx buffer **/
1241. /*******************************/
1242. {
1243. int i;
1244.
1245. txdata [0] =txpack.head;
1246. txdata [1] =txpack.pack_num;
1247. txdata[2] =txpack.pack_cmpl;
1248. for (i=0; i (128; i++)
1249.     txdata [i+3] = txpack.buffer [i];
1250. txdata [131] =txpack.chksm;
1251. }
```

231928-60

82510 XMODEM Implementation (Continued)

```
PAGE 24       MAIN PROGRAM   ftp.c  82510 XMODEM


1252.
1253. check_wait ()
1254. /*******************************************************/
1255. /**** PROCEUDRE CHECK_WAIT                        *****/
1256. /****                                             *****/
1257. /****           checks  Tx Timer, ccr_to_get and  *****/
1258. /****                   get_ccr_req  and returns: *****/
1259. /****                                             *****/
1260. /****            Time Out - Tx Timer = 0          *****/
1261. /****            rx_ACK   - Ack received          *****/
1262. /****            rx_NAK   - Nak received          *****/
1263. /****            waiting  - tx Timer not expired  *****/
1264. /****                                             *****/
1265. /****                                             *****/
1266. /*******************************************************/
1267. {
1268.
1269.  if ((!tx_time_cnt) && (get_ccr_rq ==active))        /* if tx Timer expired
1270.                                                       and still waiting
1271.                                                       for control char,then
1272.                                                       Time out */
1273.        return (time_out);
1274.  else
1275.     if (get_ccr_rq == inactive)                    /* Ctl-Char rcvd then
1276.                                                       return status */
1277.        {
1278.         switch (ccr_to_get)
1279.         {
1280.          case ACK :
1281.                return (rx_ACK);
1282.          break;
1283.
1284.          case NAK :
1285.                return (rx_NAK);
1286.          break ;
1287.
1288.       default:
1289.                return (rx_gen);                    /* corrupted ctl char */
1290.          break;
1291.         }
1292.        }
1293.     else
1294.         if ((tx_time_cnt )0) && (get_ccr_rq ==active))
1295.             return (waiting);
1296. }
1297.
1298.
```

231928-61

82510 XMODEM Implementation (Continued)

```
PAGE 25       MAIN PROGRAM   ftp.c  82510 XMODEM


1299. abort_tx ()
1300. /*********************************************************/
1301. /****                                                 *****/
1302. /****    Abort transmission, reintialize              *****/
1303. /****                          Transmitter            *****/
1304. /****                          Flags                  *****/
1305. /****                                                 *****/
1306. /*********************************************************/
1307.
1308. {
1309.         eof =false;
1310.         txrflg = mkpkt;
1311.         quit =false;
1312.         key = 0;
1313.         tx_state =tx_idle;
1314.         tx_cmd = inactive;
1315.         send_ccr_req = inactive;
1316.         tx_indx = 0;
1317.         tx_req =inactive;
1318.         ccr_to_tx = 0;
1319.         tx_byte_cnt =0;
1320.         pkts_sent =0;
1321.         tx_time_cnt =0;
1322.         get_ccr_rq =0;
1323.         ccr_to_get  =0;
1324.         set_bank (00);
1325.         outp ((bpa+1),0x27);
1326.         set_bank (001);
1327.         outp((bpa+6),0x0D);
1328.         tx_state =tx_idle;
1329.         prmsg ("transmitter reset");
1330. }
1331.
1332.
1333. wait_rx ()
1334. /*********************************************************/
1335. /****                                                 *****/
1336. /****    WAIT_RX:                                     *****/
1337. /****          checks rx timer, and returns the       *****/
1338. /****          following value :                      *****/
1339. /****               SOH - SOH received                *****/
1340. /****               EOT - EOT received                *****/
1341. /****               time out - rx timer expired       *****/
1342. /****               waiting - waiting for event       *****/
1343. /****                                                 *****/
1344. /*********************************************************/
1345.
1346. {
1347.     if ((ctl_rxd_flg == active) && (rx_time_cnt !=0))
1348.     {
1349.         ctl_rxd_flg =inactive;
1350.         return ( rx_ctl_chr);
1351.     }
1352.     else
1353.         if ( rx_time_cnt == 0)
1354.             return (time_out);
1355.         else
1356.             return (waiting);
1357. }
```

231928-62

82510 XMODEM Implementation (Continued)

```
PAGE 26       MAIN PROGRAM   ftp.c  82510 XMODEM


1358. chkpkt (pknum)
1359. /*********************************************/
1360. /**     verifies the checksum and packet    **/
1361. /**     number of the received packet       **/
1362. /**     returns a status code               **/
1363. /**                                         **/
1364. /**     EOK      - Packet Ok                **/
1365. /**     EPKNUM   - Error in packet number   **/
1366. /**     ECHKSUM  - Error in Check Sum       **/
1367. /**     EPKCMPL  - Erro in packet complement **/
1368. /**                                         **/
1369. /*********************************************/
1370. int pknum;
1371. {
1372. int i;
1373. int ckm;
1374. int sum = 0;
1375. char    cmpl,rxcmpl,cpk,chrckm;
1376.
1377.
1378.     cpk =pknum;
1379.     if (cpk ==rxdata[0])                    /* packet number correct */
1380.      {
1381.       cmpl = rxdata [0];
1382.       rxcmpl =rxdata [1];
1383.       if (rxcmpl == ~cmpl )                 /* packet number complmt */
1384.         {
1385.             for (i=2; i(130; i++)
1386.                 sum = sum +rxdata [i];
1387.             ckm = sum % 255;
1388.             chrckm = ckm;
1389.             pk_chksm =chrckm;
1390.             if (chrckm == rxdata [130])     /* checksum correct */
1391.                 return (eok);
1392.             else
1393.                 return (echksm);
1394.
1395.         }
1396.         else
1397.            return (epkcmp);
1398.      }
1399.        else
1400.        {
1401.         if ((rxdata [0] == cpk -1) && (cpk )1))    /* old packet number
1402.                                                     received */
1403.            return (eold);
1404.         else
1405.            return (epknum);
1406.         }
1407. }
```

231928-63

82510 XMODEM Implementation (Continued)

```
PAGE 27       MAIN PROGRAM   ftp.c  82510 XMODEM

1408. /*********************************************************/
1409. /****       tx_i_dis    PROCEDURE                    *****/
1410. /****   Disables txm and Tx FIFO interrupts          *****/
1411. /****   from GER register                            *****/
1412. /*********************************************************/
1413. tx_i_dis ()
1414. {
1415. mskint4 ();                              /* disable interrupts */
1416. set_bank (00);                           /* switch to bank zero */
1417. outp ((bpa+1),(inp(bpa+1) & txidb));     /* mask out interrupts in
1418.                                            GER - TxM and Tx FIFO */
1419. set_bank (01);                           /* set bank one */
1420. enbint4 ();                              /* enable interrupts */
1421.
1422.
1423.
1424.
1425.
1426. }/*********************************************************/
1427. /****          tx_i_enb    PROCEDURE                  *****/
1428. /****   enables the TXM and TX FIFO Interrupts       *****/
1429. /****   from GER register                            *****/
1430. /*********************************************************/
1431.
1432. tx_i_enb ()
1433. {
1434. mskint4 ();                              /* disable interrupts */
1435. set_bank (00);                           /* switch to bank zero */
1436. outp ((bpa+1),(inp(bpa+1) | txien));     /* enable TXM AND TX FIFO */
1437. set_bank (01);                           /* return to bank one */
1438. enbint4 ();                              /* enable interrupts */
1439.
1440. }
1441.
1442. sh_pkt_param ()
1443. /************************************************/
1444. /** Displays the parametrs of the Received     **/
1445. /** packet number, and the expected parameters **/
1446. /************************************************/
1447. {
1448. ++ bad_pkt_cnt;
1449. prmsg ("sending NAK ");
1450. printf (" error = %5u",pkst);
1451. mv_curs (13,1);
1452. printf ("exptd pkt # = %3u",exp_pkt_num);
1453. mv_curs (14,1);
1454. printf ("rxd pkt # = %3u", rxdata[0]);
1455. mv_curs (13,40);
1456. printf ("expd pkt cmpl = %#X", (~rxdata[0]));
1457. mv_curs (14,40);
1458. printf ("rxd pkt complement = %#X", rxdata[1]);
1459. mv_curs (15,1);
1460. printf ("rxd chksum = %#X", rxdata[130]);
1461. mv_curs (15,40);
1462. printf ("expd chksum = %#X",pk_chksm);
1463. send_ccr_req =active;
1464. ccr_to_tx = NAK;
1465. }
```

231928-64

82510 XMODEM Implementation (Continued)

```
1482.  else
1483.      prmsg ("file too big, cannot save in memory");
1484.  }
```

231928-65

82510 XMODEM Implementation (Continued)

```
PAGE  1        DEFINITION FILE   ftp.def  82510 XMODEM

  1. #define s1  "82510 FTP x000 6/30/86"          /* sign on message */
  2. #define bpa 0x3f8                             /* Base address */
  3. #define gir_addr    02
  4. #define esci 27                               /* escape char. in hex */
  5. #define bel  07
  6. #define msg_c 17                              /* coordinates of the
  7.                                                  message line */
  8. #define msg_r 2
  9. #define tx_c  35                             /*
 10. #define tx_r  10
 11. #define rx_c  35                                coordinates
 12. #define rx_r  12
 13. #define so_c  50
 14. #define so_r  24                             */
 15. #define false  0
 16. #define true   1
 17. #define active 1
 18. #define inactive 0
 19. #define ctl_chr 2                            /* control char transmit */
 20. #define pkt 1                                /* send packet */
 21. #define eok 5555                             /* packet received ok */
 22. #define echksm 5500                          /* checksum error */
 23. #define epkcmp  5501                         /* packet compl incorrect */
 24. #define eold    5502                         /* old pack num received */
 25. #define epknum 5503                          /* invalid packet # rcvd. */
 26.
 27. /***************************/
 28. /*** tx state defintions***/
 29. /***************************/
 30.
 31. #define tx_idle     000
 32. #define wait_NAK    001
 33. #define TO_err_60   002
 34. #define tx_rdy      003
 35. #define tx_packet   004
 36. #define wait_CC     005
 37. #define tx_pk_comp  006
 38. #define to_err      007
 39. #define txen        0x02
 40. #define mkpkt       111                      /* Transmit packet stages */
 41. #define txmtg       112
 42. #define retx        113
 43. #define waiting     114
 44.
 45. /***************************/
 46. /**  rx state definition **/
 47. /***************************/
 48.
 49. #define rx_idle     000
 50. #define rx_rdy      001
 51. #define rx_pkt      002
 52.
 53.
 54. /***************************/
 55.
 56. #define     time_out    90                   /* rx state signal values */
 57. #define     rx_NAK      91
 58. #define     rx_ACK      92
 59. #define     rx_gen      93
 60.
```

231928-66

82510 XMODEM Implementation (Continued)

```
PAGE  2        DEFINITION FILE   ftp.def  82510 XMODEM


 61.
 62. /*********************************/
 63. /**     Protocol Control     ****/
 64. /**     characters           ****/
 65. /*********************************/
 66.
 67.
 68. #define NAK      0x14                    /* Negative Ack */
 69. #define ACK      0x06                    /* Positive Ack */
 70. #define SOH      0x01                    /* Start of Header */
 71. #define EOT      0x04                    /* End of Text */
 72. #define CAN      0x18
 73. #define NUL      0x00
 74.
 75. /*********************************/
 76. /**      interrupt source     ****/
 77. /*********************************/
 78.
 79. #define timer    05                      /* 82510 int. vectors */
 80. #define txm      04
 81. #define ccr      03
 82. #define rxf      02
 83. #define txf      01
 84. #define txien    0x12                    /* unmask TxM and Tx FIFO */
 85. #define txidb    0x2D                    /* mask  TxM and Tx FIFO */
 86. #define ccien    0x04                    /* enable CCR interrupts */
 87. #define ccidb    0x33                    /* mask CCR int */
 88. #define blkenb   0x25                    /* enable, block interrupts
                                                 through GER for 82510 */
 89.
 90.
 91. /*******************************/
 92. /**      8259A values *********/
 93. /*******************************/
 94.
 95. #define eoi      0x20                    /* end of interrupt */
 96. #define ip00     0x20                    /* 8259A port 0 */
 97. #define ip01     0x21                    /* 8259A port 1 */
```

231928-67

82510 XMODEM Implementation (Continued)

```
PAGE  1       CRT I/O ROUTINES    cio.c   82510 XMODEM

  1. #include "ftp.def"
  2.
  3. CLR()
  4. /*********************************************************/
  5. /****                                                 *****/
  6. /****     PROCEDURE CLR                               *****/
  7. /****                                                 *****/
  8. /****                  clears screen                  *****/
  9. /****                                                 *****/
 10. /****                                                 *****/
 11. /****                                                 *****/
 12. /*********************************************************/
 13.
 14. {
 15. int escchr = esci;
 16.
 17.     putch (escchr);
 18.     printf ("[2J");
 19.
 20. }
 21.
 22.
 23. VOFF ()
 24. /*********************************************************/
 25. /****                                                 *****/
 26. /****        PROCEDURE VOFF                           *****/
 27. /****                                                 *****/
 28. /****                Turns Reverse Video OFF          *****/
 29. /****                                                 *****/
 30. /****                                                 *****/
 31. /****                                                 *****/
 32. /*********************************************************/
 33.
 34. {
 35. int escchr = esci;
 36.
 37.     putch (escchr);
 38.     printf ("[0m");
 39.
 40. }
 41.
 42.
 43. RVON ()
 44. /*********************************************************/
 45. /****                                                 *****/
 46. /****    PROCEDURE RVON                               *****/
 47. /****                                                 *****/
 48. /****              Reverse Video ON                   *****/
 49. /****                                                 *****/
 50. /****                                                 *****/
 51. /****                                                 *****/
 52. /*********************************************************/
 53.
 54. {
 55. int escchr = esci;
 56.
 57.     putch (escchr);
 58.     printf ("[7m");
 59. }
 60.
```

231928-68

82510 XMODEM Implementation (Continued)

```
PAGE  2        CRT I/O ROUTINES    cio.c   82510 XMODEM

 61. OPEN_WIND (row,col,stg)
 62. int     row;
 63. int     col;
 64. char    stg[];
 65. /*************************************************************/
 66. /****                                                     *****/
 67. /****   PROCEDURE OPEN_WIND                               *****/
 68. /****                                                     *****/
 69. /****            prints a string in reverse video         *****/
 70. /****            at the given location                    *****/
 71. /****                                                     *****/
 72. /****                                                     *****/
 73. /*************************************************************/
 74.
 75. {
 76.
 77.     MV_CURS (row, col);
 78.     RVON ();
 79.     printf ("%s",stg);
 80.     VOFF();
 81.
 82. }
 83. BEEP ()
 84. /*************************************************************/
 85. /****                                                     *****/
 86. /****  PROCEDURE BEEP                                     *****/
 87. /****                                                     *****/
 88. /****             produces a beep                         *****/
 89. /****                                                     *****/
 90. /****                                                     *****/
 91. /****                                                     *****/
 92. /*************************************************************/
 93.
 94. {
 95. int belchr = bel;
 96.
 97.     putch (belchr);
 98.
 99. }
100.
101. CLL(row,col)
102. int row;
103. int col;
104. /*************************************************************/
105. /****                                                     *****/
106. /****   PROCEDURE CLL                                     *****/
107. /****                                                     *****/
108. /****             clear line at given coordinate          *****/
109. /****                                                     *****/
110. /****                                                     *****/
111. /****                                                     *****/
112. /*************************************************************/
113.
114. {
115. int escchr = esci;
116.     MV_CURS (row, col);
117.     putch (escchr);
118.     printf ("[K");
119. }
120.
```

231928-69

82510 XMODEM Implementation (Continued)

```
PAGE  3        CRT I/O ROUTINES    cio.c   82510 XMODEM

121.
122.
123. CLMS()
124. /*********************************************************/
125. /****                                                 *****/
126. /****       PROCEDURE CLMS                            *****/
127. /****                                                 *****/
128. /****              clear message line                 *****/
129. /****                                                 *****/
130. /****                                                 *****/
131. /****                                                 *****/
132. /*********************************************************/
133.
134. {
135.     CLL (msg_r,msg_c);
136. }
137.
138. prmsg (msg)
139. char    msg [];
140. /*********************************************************/
141. /****                                                 *****/
142. /****   PRINTS MESSAGE AT MESSAGE LINE                *****/
143. /****                                                 *****/
144. /****                                                 *****/
145. /****                                                 *****/
146. /****                                                 *****/
147. /****                                                 *****/
148. /*********************************************************/
149.
150. {
151.     clms ();
152.     printf (" %s", msg);
153. }
154.
155. CLLC ()
156. {
157. int escchr = esci;
158.     putch (escchr);
159.     printf ("[K");
160. }
161.
162. MV_CURS (x,y)
163. /*********************************************************/
164. /****                                                 *****/
165. /****  PROCEDURE MV_CURS                              *****/
166. /****                                                 *****/
167. /****              moves cursor to specified          *****/
168. /****              location.                          *****/
169. /****                                                 *****/
170. /*********************************************************/
171.
172. int x;
173. int y;
174. {
175. int escchr = esci;
176.
177.     putch (escchr);
178.     cprintf ("[%u;%uH",x,y);
179. }
```

231928-70

82510 XMODEM Implementation (Continued)

```
PAGE  1     ASM 86 INTERRUPT INIT. ih1.asm 82510 XMODEM

  1. NAME        ftpih
  2.
  3. DGROUP  GROUP   DATA
  4. DATA    SEGMENT WORD PUBLIC 'DATA'
  5.     ASSUME  DS:DGROUP
  6. DATA    ENDS
  7.
  8. EXTRN       isr_510:far
  9.
 10. _PROG   SEGMENT BYTE PUBLIC 'PROG'
 11.     ASSUME  CS:_PROG
 12.
 13. PUBLIC      init_ih
 14. PUBLIC      ih510
 15.
 16. init_ih    PROC    far
 17.     push    BP
 18.     push    DX
 19.     push    AX
 20.     push    DS
 21.     mov     DX, OFFSET ih510
 22.     push    CS
 23.     pop     DS
 24.     mov     AH,25H          ;DOS vector setup call
 25.     mov     AL,0CH          ;COM1  vector
 26.     INT     21H             ;DOS system call
 27.     pop     DS
 28.     pop     AX
 29.     pop     DX
 30.     pop     BP
 31.     ret
 32. init_ih    ENDP
 33.
 34. ih510      PROC    far
 35.     push    BP
 36.     push    AX
 37.     push    BX
 38.     push    CX
 39.     push    DX
 40.     push    SI
 41.     push    DI
 42.     push    DS
 43.     push    ES
 44.     mov     AX, DGROUP
 45.     mov     DS, AX
 46.     call    isr_510
 47.     pop     ES
 48.     pop     DS
 49.     pop     DI
 50.     pop     SI
 51.     pop     DX
 52.     pop     CX
 53.     pop     BX
 54.     pop     AX
 55.     pop     BP
 56.     iret
 57. ih510      ENDP
 58.
 59. _PROG   ENDS
 60. end
```

231928-71

82510 XMODEM Implementation (Continued)

June 1987

2

High Performance Driver for 82510

DAN GAVISH and TSVIKA KURTS
SYSTEM VALIDATION

Order Number: 292038-001

HIGH PERFORMANCE DRIVER FOR 82510

CONTENTS

PAGE

1.0 OVERVIEW 2-265

2.0 INTRODUCTION 2-265

2.1 CPU Load Consideration 2-265

2.2 82510 Features Used In This Implementation 2-265

3.0 THE BURST ALGORITHM 2-266

3.1 Background 2-266

3.2 Burst Algorithm Description 2-266

4.0 SOFTWARE MODULE MAP 2-267

5.0 HARDWARE VEHICLE DESCRIPTION 2-268

6.0 SOFTWARE MODULE DESCRIPTIONS 2-268

6.1 MAIN 2-268

6.2 The Burst Algorithm Modules 2-269

6.3 Initializations 2-271

6.4 Interrupt Handler 2-275

APPENDIX A—PL/M SOURCE FILE .. 2-280

APPENDIX B—82510 BASED SBX SERIAL CHANNEL 2-293

1.0 OVERVIEW

The 82510 Asynchronous Serial Controller is a CHMOS UART which provides high integration features to offload the host CPU and to reduce the system cost.

This Ap-Note presents a mechanism for reduction and optimization of interrupt handling during asynchronous communication using the 82510. The mechanism is valuable in applications where handling of interrupts degrades system performance i.e., when high baud rate is used, when multiple channels are handled or whenever real-time constraints exist. This implementation of the mechanism is a software driver that transmits or receives characters at 288000 bits per second.

The driver is based on the burst algorithm which uses the 82510 features (FIFOs, Timers, Control Character Recognition etc.) to reduce CPU overhead. CPU is significantly off-loaded for other tasks — about 75% of the usual load is saved.

The driver can be easily modified to run in conjunction with other 82510 features such as the MCS-51 9-bit Protocol.

This document provides a full description of the driver. The burst algorithm is presented in Section 3, the software module flow-charts and their descriptions are presented in Section 6, and the PL/M software listing is given in Appendix A.

2.0 INTRODUCTION

2.1 CPU Load Consideration

The trend towards multi-tasking systems, combined with higher baud rates and increasing the number of channels per CPU, has led to the need for decreasing the CPU bandwidth consumed by the async communications for each byte transfer. Whenever the CPU is interrupted, a certain amount of CPU time is lost in implementing the context switch. This overhead can be as high as hundreds of microseconds per interrupt, depending on the specific operating system parameters. Thus, in high baud-rate or multi-channel environments, where the interrupt frequency is very high, a substantial portion of the CPU time is taken up by this interrupt overhead. Therefore, systems usually require minimization of the number of interrupt events. In the case of an asynchronous communication channel, reduction of the number of interrupts can be achieved by servicing (i.e., transferring to/from the buffer) as many characters as possible whenever the interrupt routine is activated. This can be done by utilizing FIFOs to hold received or transmitted characters, so that the CPU is interrupted only after a certain number of characters have been received or transmitted. Using a receive FIFO may cause a potential problem: Due to the random rate of character arrival in asynchronous communications, there is a chance that characters will be "trapped" in the Rx FIFO for extended periods of time. In order to avoid such situations, a Rx FIFO time-out mechanism can be implemented using the 82510 timer. The time-out indicates that a certain amount of time has elapsed since the last read operation was performed. It causes the CPU to check the Rx FIFO and read any characters that are present. This process, however, introduces the additional overhead of the timer interrupt. This Ap-Note describes the use of the burst algorithm to avoid the timer interrupt overhead while maintaining the use of the Rx FIFO.

2.2 82510 Features Used In This Implementation

The following new 82510 features were used in this implementation:

2.2.1 FIFOs

The 82510 is equipped with 2 four-byte FIFOs, one for reception and one for transmission. While characters are being received, a Rx FIFO interrupt is generated, when the Rx FIFO occupancy increases above a programmable threshold. While characters are being transmitted, a Tx FIFO interrupt is generated, when the Tx FIFO occupancy drops below a programmable threshold. The two thresholds are software programmable, for maximum optimization to the system requirements.

2.2.2 TIMER

The 82510 is equipped with two on chip timers. Each timer can be used as a baud rate generator or as a general purpose timer. When two independent baud rates are required for transmit and receive, the two timers can be used to generate both baud rates internally. Otherwise, one timer can be used for external purposes. The timer is loaded with its initial value by a software command and it counts down using system clock pulses. When it expires, a maskable interrupt is generated.

2.2.3 CONTROL CHARACTER RECOGNITION

Depending on the application, the software usually checks the received characters to determine whether certain control characters have been received, in which case special processing is performed. This loads the CPU, as every received character should be compared to a list of control characters. With the 82510, the CPU is offloaded from this overhead. Every received character is checked by the 82510, and compared to either a standard set of control characters (ASCII or EBCDIC) or to special user defined control characters. The software does not need to check the received characters, and a special interrupt is provided when a received control character is detected by the 82510. The specific operation mode (standard set, user defined, etc.) is programmable.

2.2.4 INTERRUPT CONTROLLING MECHANISM

The twenty possible interrupt sources of the 82510 are grouped into six blocks: Timer, Tx machine, Rx machine, Rx FIFO, Tx FIFO, or Modem. Interrupt source blocks are prioritized. The interrupt management is performed by the 82510 hardware. The CPU is interrupted by a single 82510 interrupt signal. The interrupt handler is reported on the highest priority pending interrupt block (GIR) and on all the pending interrupt blocks (GSR), as well as on the specific interrupt source. Interrupts are maskable at the block level and source level. Interrupts can be automatically acknowledged (become not pending) when serviced by the software, or manually acknowledged by an explicit command.

3.0 THE BURST ALGORITHM

3.1 Background

The 82510 FIFOs are used to reduce the CPU interrupt load. When a burst of characters is transmitted or received, the CPU is interrupted only once per transmission or reception of up to four characters. FIFO thresholds are programmable; thus, when high system interrupt latency is expected, an optimal threshold may be selected for the desired trade-off between the CPU load, and the acceptable system interrupt latency. The required Rx FIFO threshold is also a function of the receive character rate. When the rate is high, a deep FIFO is required. When the rate is very low (e.g., hundreds of milliseconds between characters), a low threshold is needed, to reduce the maximum character service latency (a character is available to the application program only after it is stored in the receive buffer).

The software mechanism described here tunes the Rx FIFO threshold dynamically when the incoming character rate is variable. The algorithm uses one of the 82510 on-chip timers for time measurement, in order to automatically adapt the threshold to the character reception rate. This is done without loading the CPU with the overhead of serving excessive interrupts generated by the timer mechanism itself.

3.2 Burst Algorithm Description

The 82510 timer is initialized to the time-out value with every Rx FIFO interrupt. The time-out value is the maximum acceptable time between a character's reception and its storage in the receive buffer, but not less than five character-times. Upon reception of the next character, the timer status is examined to determine whether the character rate is high (the timer has not yet expired) or low (the timer has expired).

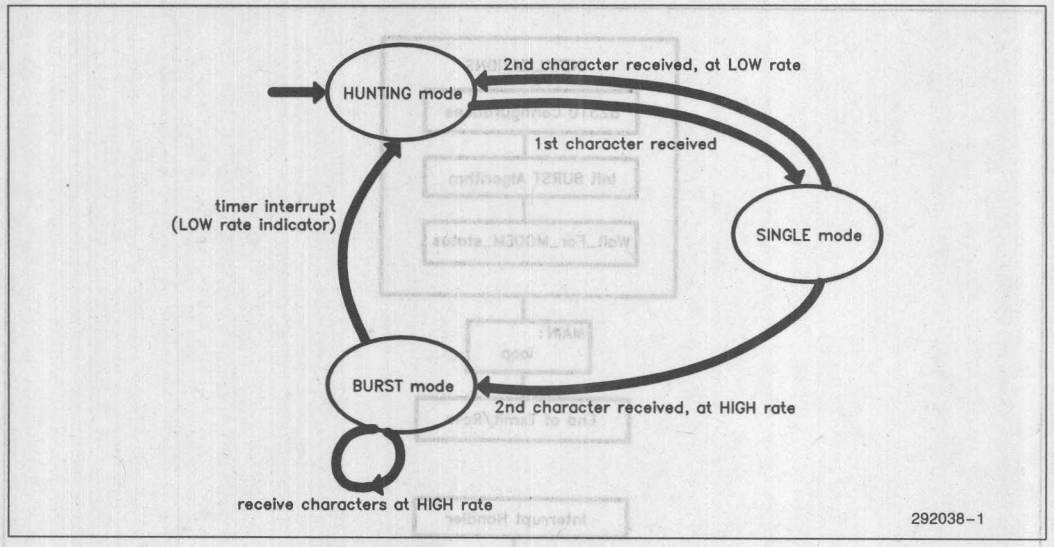


Figure 1. Burst Algorithm State Diagram

The algorithm is best described as a finite state machine that can be in one of three modes: HUNTING mode, SINGLE mode, or BURST mode. In HUNTING mode, after the first character received interrupts the CPU, the mode switches to SINGLE. On receiving a character in SINGLE mode (that is the second character) the timer is examined; if the character rate is very low, the mode is switched back to HUNTING. Otherwise, the rate is high enough to switch to BURST mode. In BURST mode, the Rx FIFO threshold is maximal. The machine remains in BURST mode as long as a burst of characters is being received. When the rate of character reception becomes low, the timer eventually expires generating a timer interrupt which switches the mode back to HUNTING.

Note that while a burst of characters is being received, the CPU is interrupted only once per four received characters. If the characters are received at a very low rate, an interrupt occurs for each received character. The CPU is interrupted by the timer only once, when the burst terminates. See Figure 1 for a state diagram.

For more details about the burst algorithm see paragraph 6.2.

4.0 SOFTWARE MODULE MAP

The driver contains the following software modules:

- MAIN
- BURST ALGORITHM
  - — Burst Algorithm Initialization (*)
  - — Rx FIFO Step (*)
  - — HUNTING mode
  - — SINGLE mode
  - — BURST mode
  - — Timer Step (*)
- INITIALIZATIONS
  - — Wait for Modem Status
- INTERRUPT HANDLER
  - — Rx FIFO Interrupt Service Routine
  - — Tx FIFO Interrupt Service Routine
  - — Status Interrupt Service Routine
  - — Timer Interrupt Service Routine
  - — Modem Interrupt Service Routine

(*) The burst algorithm modules are called by the initialization module and by the interrupt handler modules.

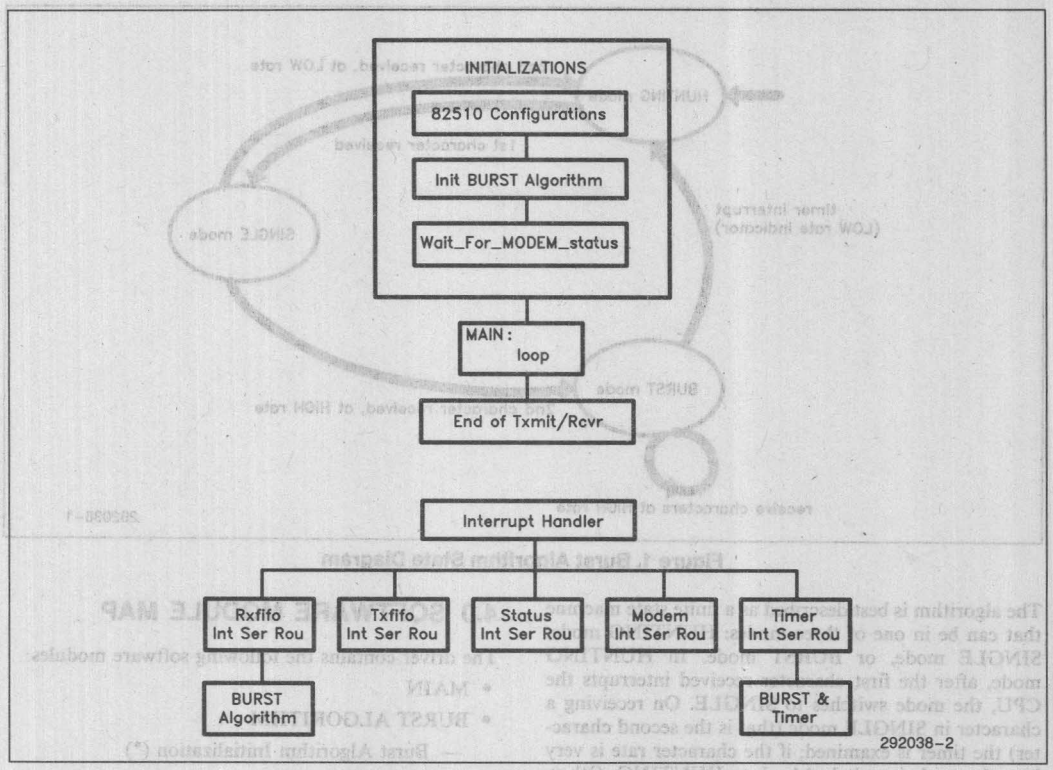


Figure 2. Modules Block Diagram

5.0 HARDWARE VEHICLE DESCRIPTION

The driver was tested at 288000 baud, on an 80186 based system, with an 8 MHz local bus running with 2 wait-states, and an 18.432 MHz 82510 clock. Two stations were involved: one transmitter station and one receiver station. Each station consisted of an iSBC186/51 with a 82510 based SBX board connected to it. See Appendix B for description of the SBX board.

This driver is, nonetheless, suitable for running in a large number of system environments.

6.0 SOFTWARE MODULE DESCRIPTIONS

6.1 MAIN

The MAIN module is a simple example of an application program that uses the driver.

The communication is done between two stations: One station is the transmitter and the other one is the receiver. After interrupts are enabled, the program waits for the Finish_Tx flag or the Finish_Rx flag (for the transmitter or receiver station, respectively) to be set. In the transmitter station, the driver is preloaded with the transmit data. In the receiver station, the received data is displayed after data reception is complete.

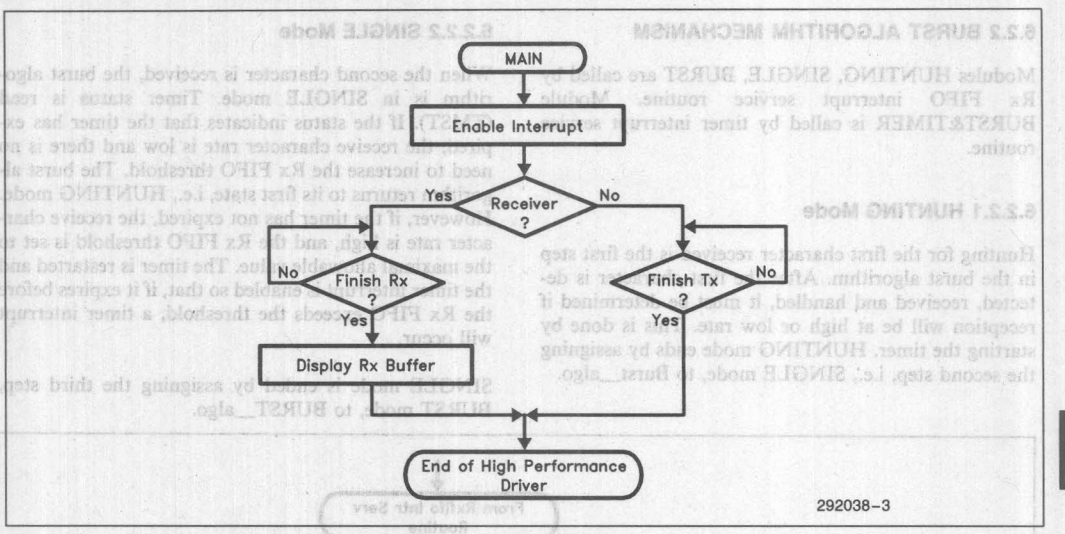


Figure 3. MAIN

6.2 The Burst Algorithm Modules

6.2.1 INITIALIZE THE BURST ALGORITHM

This module is called by the initialization module.

The global variable Burst_algo is used to indicate the current burst algorithm mode.

The burst algorithm is most useful at a baud rate of 9600 or higher. At lower baud rates, where the Rx interrupt rate is very low, the burst algorithm is degenerated (Low_baud is assigned to Burst_algo). At a baud rate of 9600 or more, the burst algorithm mechanism is initialized and starts by disabling the timer interrupt.

The initial state of the burst algorithm is HUNTING mode. In this mode, it is looking for (hunting) the first character. The Rx FIFO threshold is zero, thus the first character received interrupts CPU. This interrupt starts the burst algorithm mechanism.

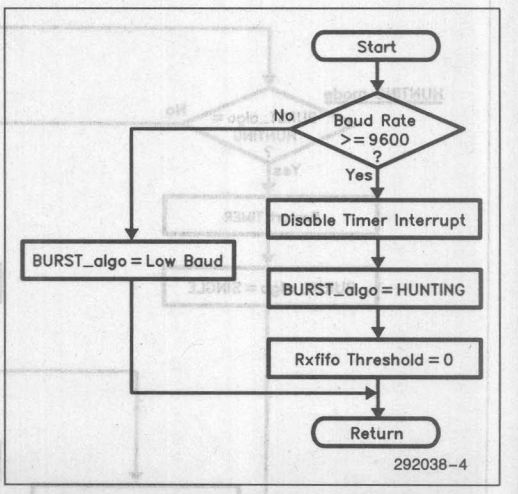


Figure 4. Initialize The Burst Algorithm

6.2.2 BURST ALGORITHM MECHANISM

Modules HUNTING, SINGLE, BURST are called by Rx FIFO interrupt service routine. Module BURST&TIMER is called by timer interrupt service routine.

6.2.2.1 HUNTING Mode

Hunting for the first character received is the first step in the burst algorithm. After the first character is detected, received and handled, it must be determined if reception will be at high or low rate. This is done by starting the timer. HUNTING mode ends by assigning the second step, i.e., SINGLE mode, to Burst_algo.

6.2.2.2 SINGLE Mode

When the second character is received, the burst algorithm is in SINGLE mode. Timer status is read (TMST). If the status indicates that the timer has expired, the receive character rate is low and there is no need to increase the Rx FIFO threshold. The burst algorithm returns to its first state, i.e., HUNTING mode. However, if the timer has not expired, the receive character rate is high, and the Rx FIFO threshold is set to the maximal allowable value. The timer is restarted and the timer interrupt is enabled so that, if it expires before the Rx FIFO exceeds the threshold, a timer interrupt will occur.

SINGLE mode is ended by assigning the third step, BURST mode, to BURST_algo.

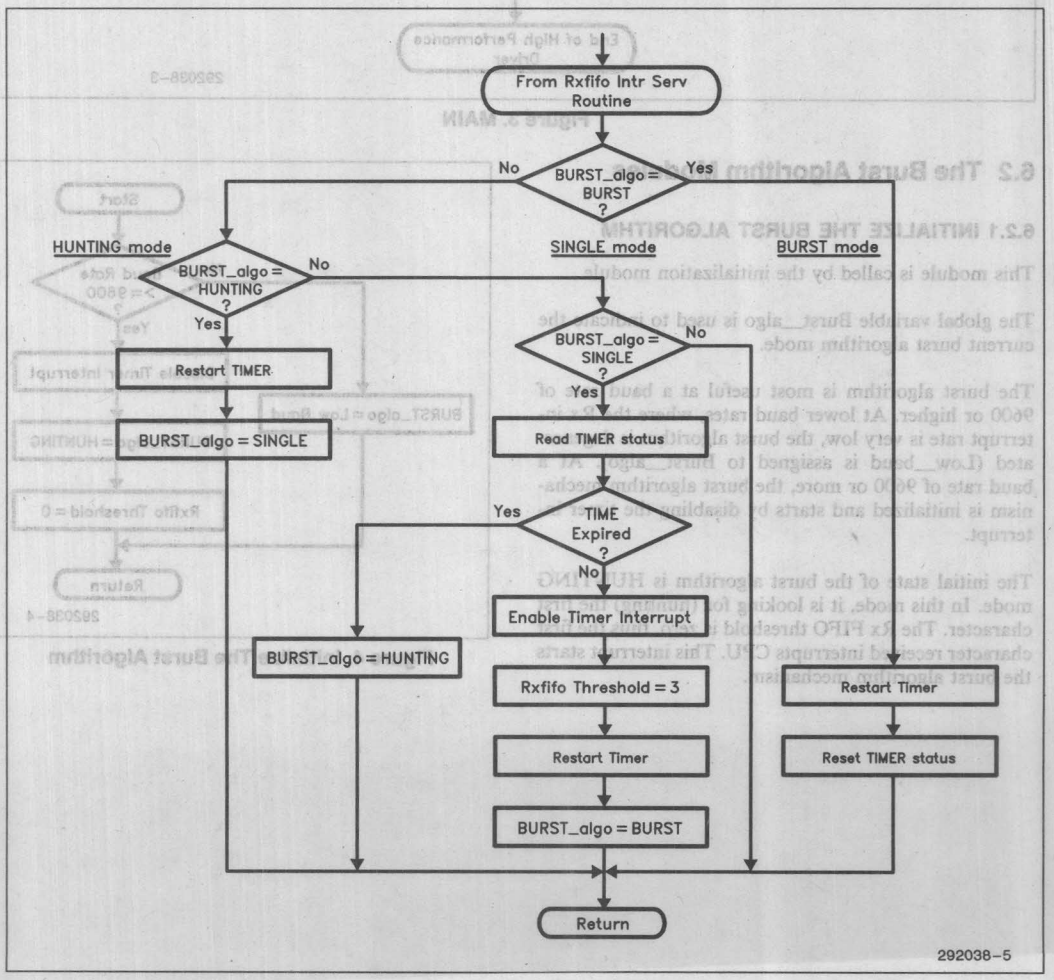


Figure 5. The Burst Algorithm

6.2.2.3 BURST Mode

The algorithm enters BURST mode as soon as the receive character rate is evaluated as high, i.e., when two successive characters are received without a timer expiration. The FIFO is now working at full threshold and the timer is used as a timeout watch dog. BURST mode is the most time-critical path of the algorithm. Therefore, it consumes a minimum amount of real time.

The timer is restarted, in order to restart a new timeout measurement. The timer status is read to trigger automatic reset of the previous status; this is done to avoid the timer interrupt if the timer has expired during the Rx FIFO interrupt service routine execution.

6.2.2.4 Timer Interrupt and Bust Algorithm

If the character reception rate becomes low, then the time between two successive Rx FIFO interrupts increases. Hence, a reduction in the reception rate causes the timeout to expire, and a timer interrupt occurs. This drives the algorithm back to HUNTING mode. The timer interrupt is disabled and the Rx FIFO threshold is configured to zero, to issue an Rx interrupt on the first hunted character.

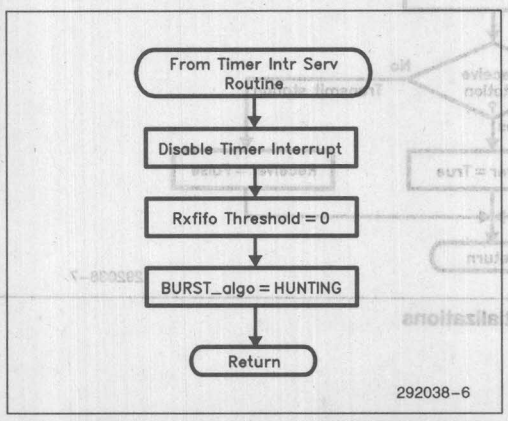


Figure 6. Timer Interrupt and BURST Algorithm

Table 1. BURST Algorithm Modes

Mode	FIFO Threshold	Timer	Timer-Interrupt
Hunting	0	Idle	Disabled
Single	0	Started	Disabled
Burst	Max.	Restarted	Enabled

6.2.3 FLOWCHART DESCRIPTION

The Rx FIFO interrupt handler executes the burst algorithm immediately after the Rx FIFO is emptied (to avoid an overrun error). The module was designed to minimize the CPU overhead inherent in the burst algorithm itself.

BURST mode is assigned the fastest path because it is the most real time sensitive mode.

SINGLE mode has a slightly longer path. However, under a high reception rate, the algorithm passes SINGLE mode once only and then stays in BURST mode until the end of the burst. Under a low reception rate the algorithm passes SINGLE mode many times, but, since the period between two successive Rx interrupts is long, this hardly affects system performance.

6.3 Initializations

This module initializes the driver. It is called at program start-up.

The 82510 is configured for the specific operation mode by the CONFIG__82510 submodule: A Software Reset command is issued, and then the character configuration is selected. In the receiver station ACR0 and ACR1 Registers are loaded with the End-Of-File ASCII character, so that the Control Character Recognition feature of the 82510 can be used to detect the specific file terminator. In the transmitter station, the ASCII characters XOFF and XON are loaded to ACR0 and ACR1, respectively, to detect transmit-off/on requests automatically. The use of the control character recognition feature of the 82510 reduces system overhead, as the software does not need to check every received character. A special interrupt is received when the 82510 hardware detects a received control character.

Interrupt sources are enabled (note that a Tx interrupt will occur immediately). BRGA is loaded to generate the required baud rate (288000 baud in this specific implementation). Rx FIFO depth is set to 4. The Tx and Rx FIFO thresholds are initialized to 0. BRGB is selected to function as a timer, and is loaded with the timeout value (7 ms at 18.432 MHz, in this implementation). The RxC and TxC sources are selected to be BRGA.

The burst algorithm parameters are initialized by INIT__BURST. WAIT__FOR__MODEM__STATUS is called and implements a wait until the modem handshake DSR signal is set. If WAIT__FOR__MODEM__STATUS returns with a timeout error, the modem error is processed. If no error has occurred, the following parameters are initialized: Finish__Rx and Finish__Tx flags, receive and transmit buffer pointers, and the receiver flag. All status registers are cleared by issuing a STATUS CLEAR command to the ICM register.

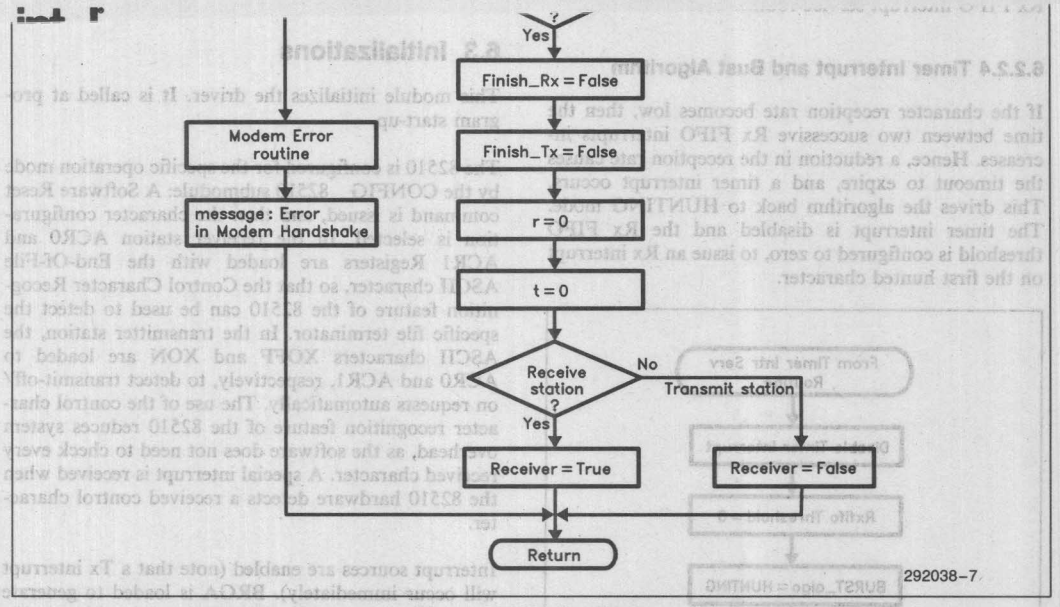


Figure 7. Initializations

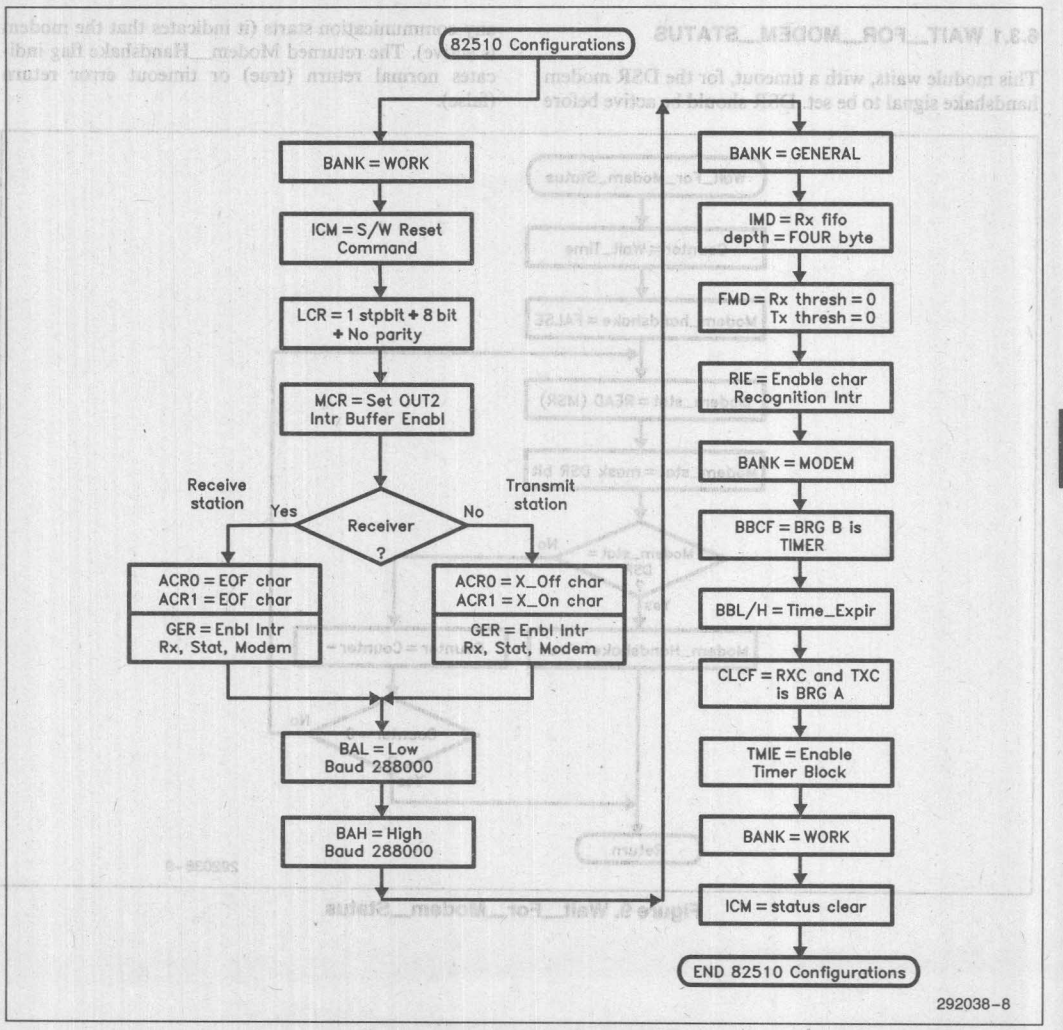


Figure 8. 82510 Configurations

6.3.1 WAIT_FOR_MODEM_STATUS

This module waits, with a timeout, for the DSR modem handshake signal to be set. DSR should be active before any communication starts (it indicates that the modem is active). The returned Modem_Handshake flag indicates normal return (true) or timeout error return (false).

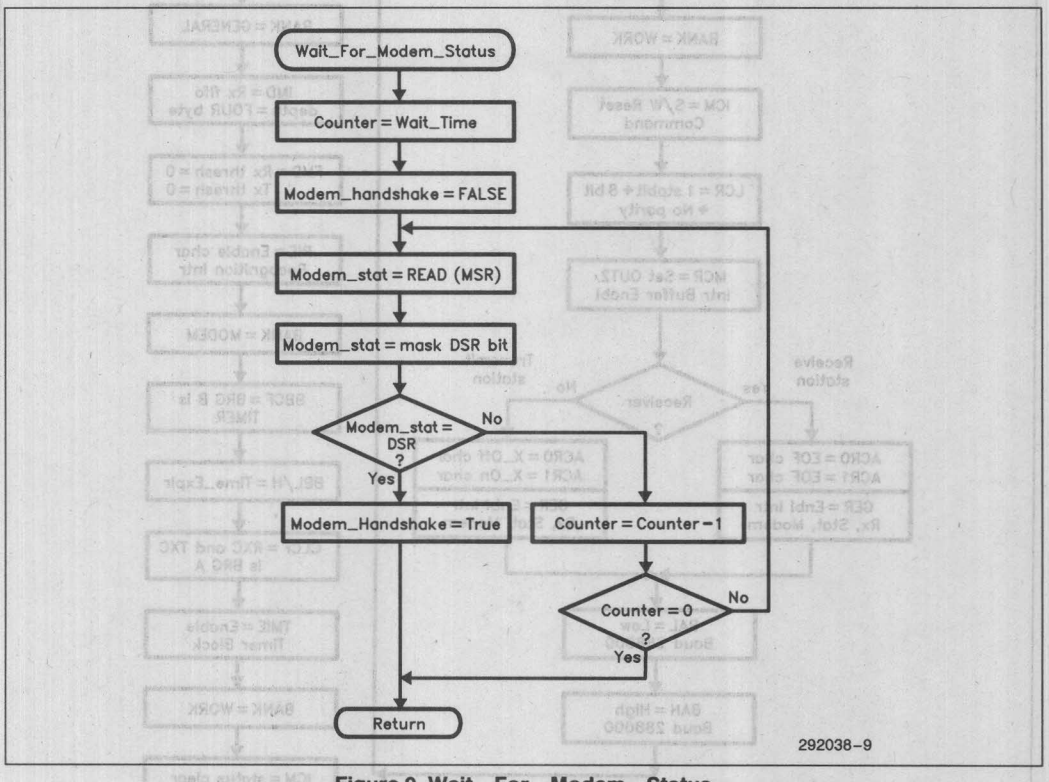


Figure 9. Wait_For_Modem_Status

6.4 Interrupt Handler

The interrupt handler services the 82510 interrupt sources. Since this is a time-critical path, the code is optimized to minimize real time consumption.

The interrupt handler services only one interrupt source at a time. This prevents CPU resource starvation from other interrupt driven devices. Interrupts are enabled at the beginning of the interrupt handler, so that higher priority interrupt sources are not disabled by the 82510 interrupt handler.

6.4.1 INTERRUPT HANDLER STRUCTURE

The interrupt handler identifies the highest priority pending 82510 interrupt, by reading GIR. The interrupt handler was designed so that shorter paths are assigned to more real time sensitive interrupt sources. Rx FIFO interrupt is the most sensitive, Tx FIFO is the second most sensitive, and so on.

The programmable interrupt controller (8259A) is assumed to be configured to "edge triggering mode" and "non-automatic end of the interrupt" mode.

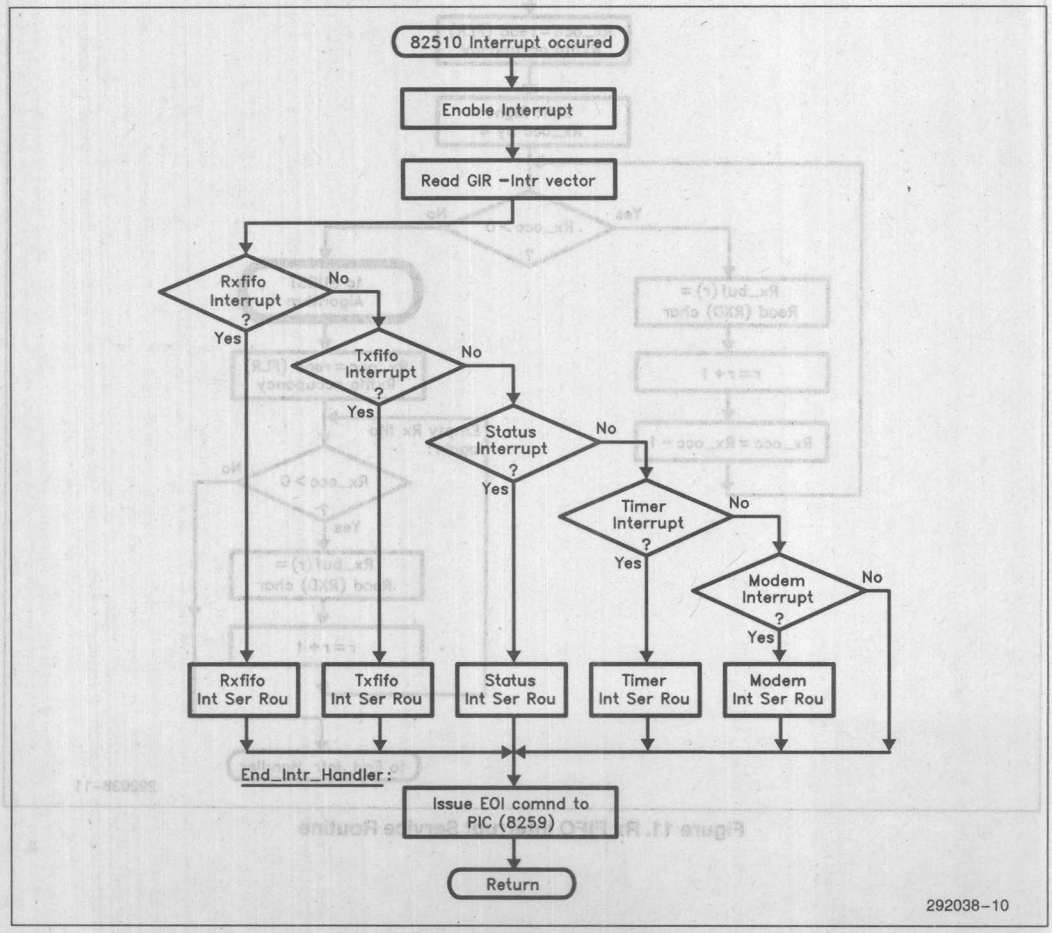


Figure 10. Interrupt Handler

6.4.2 Rx FIFO INTERRUPT SERVICE ROUTINE

The Rx FIFO interrupt service routine first empties the Rx FIFO. The receive data register (RXD) is read, as many times as indicated by the FIFO occupancy register (FLR), and the characters are stored in Rx_Buf.

After emptying the Rx FIFO, the Rx FIFO interrupt service routine executes the burst algorithm (see paragraph 6.2.2). Before leaving the Rx FIFO interrupt service routine, the FIFO occupancy register is rechecked, to empty the Rx FIFO of characters that may have been received during the Rx FIFO interrupt service routine itself. This can happen if the Rx FIFO interrupt service routine has been interrupted by a higher priority interrupt.

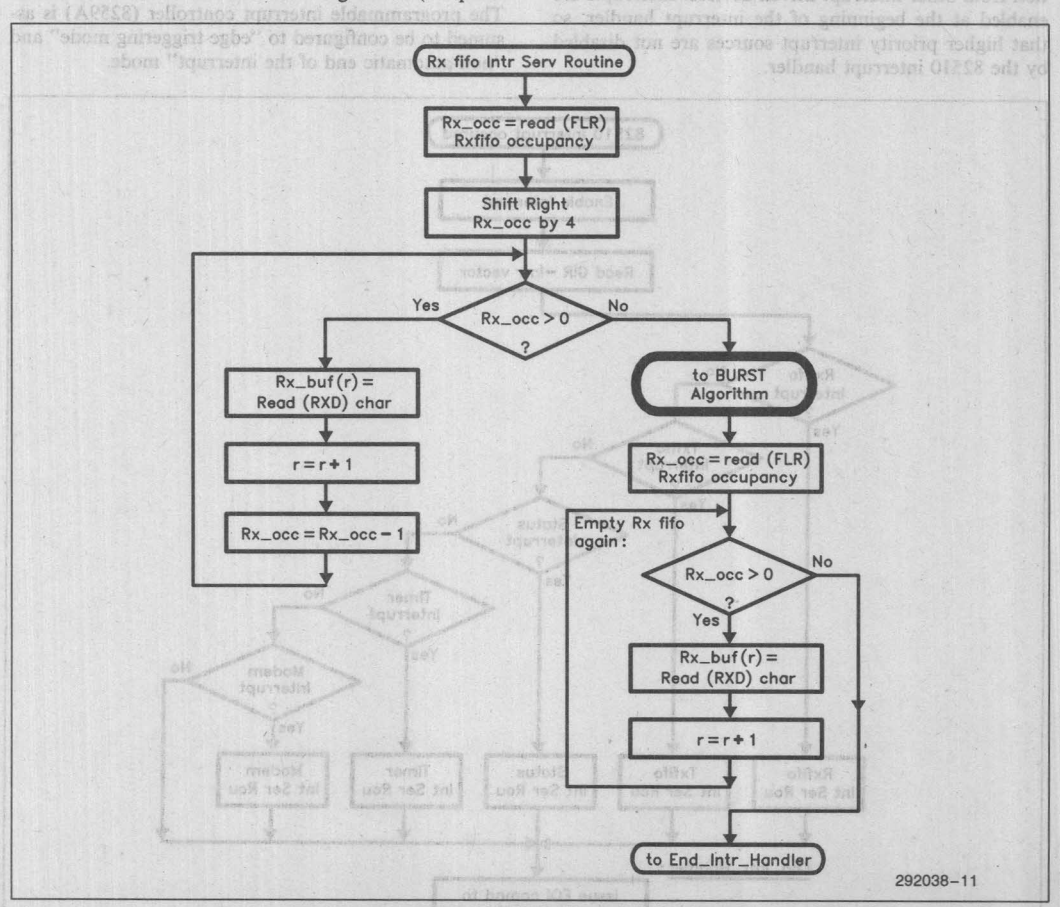


Figure 11. Rx FIFO Interrupt Service Routine

6.4.3 Tx FIFO INTERRUPT SERVICE ROUTINE

The Tx FIFO interrupt service routine fills the Tx FIFO with transmit characters while checking for the End-Of-File terminator. According to the FIFO occupancy register (FLR), the Tx FIFO is loaded (by writing to TXD) until it is full or until the End-Of-File character is detected. The transmitted characters are taken from Tx_Buf. If an End-Of-File character is identified, then the transmission is immediately ended by disabling all 82510 interrupts and setting the Finish_Tx flag.

6.4.4 STATUS INTERRUPT SERVICE ROUTINE

The status interrupt service routine has four objectives:

— To empty the Rx FIFO.

— To stop reception if an End-Of-File character is identified by the control character recognition mechanism (in the receiver station).

— To disable or enable the Tx interrupt if a XOFF or XON character, respectively, is identified by the control character recognition mechanism (in the transmitter station).

— To handle parity, framing, or overrun errors (in the receiver station).

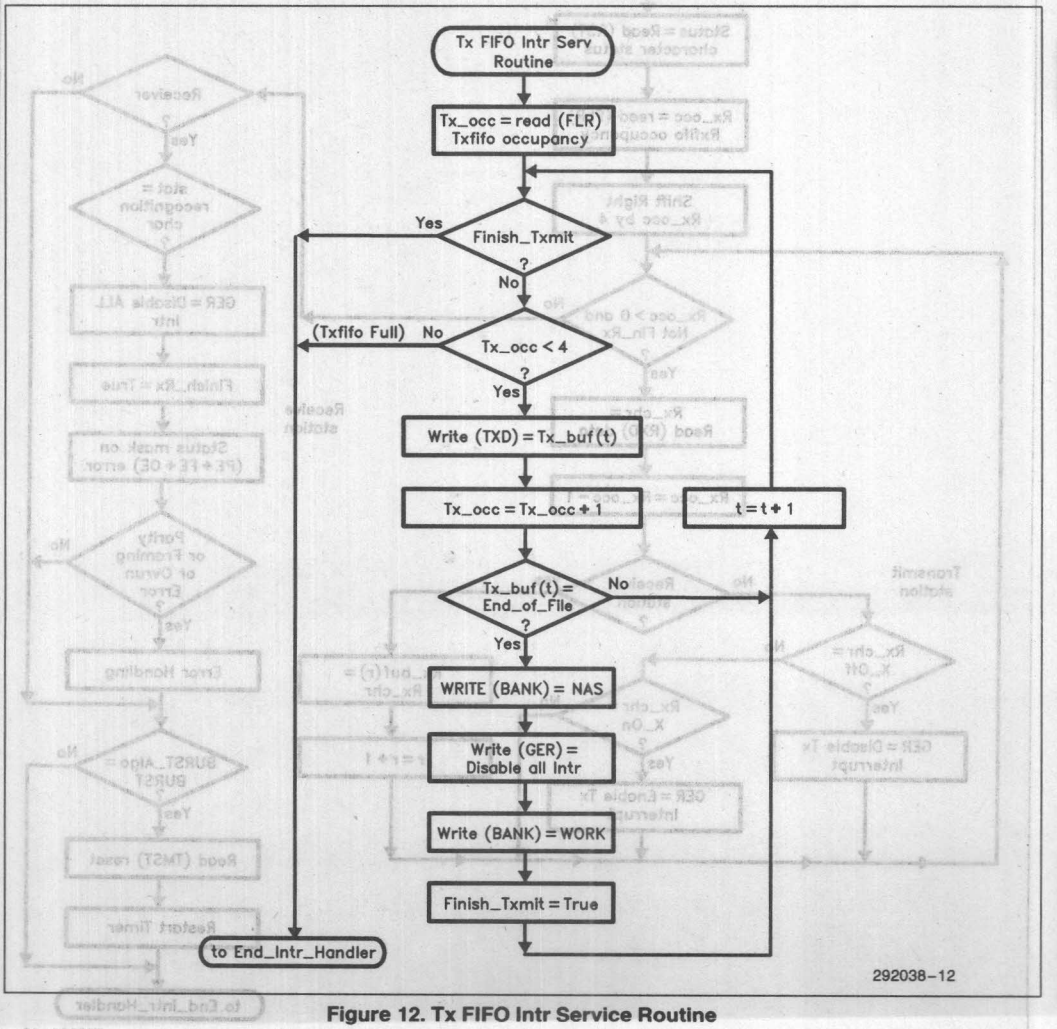


Figure 12. Tx FIFO Intr Service Routine

First the Rx FIFO is emptied. In the receiver station, the RST register is checked to determine whether an End-Of-File terminator has been identified by the 82510, in which case reception is stopped immediately by disabling all interrupt sources and setting the Finish_Rx flag. In the transmitter station, the received characters are checked to identify the received control character. If XOFF is identified, Tx interrupt is disabled. If XON is identified, Tx interrupt is enabled. Note that the software does not need to check for any control character during normal reception; the control characters are identified by the 82510 device.

RST is checked for parity, framing or overrun errors. If one of these errors has occurred, then the error handling routine is executed.

If status interrupt occurs while Burst_algo is assigned to BURST mode, the timer is restarted.

Note that status interrupt is enabled at both stations.

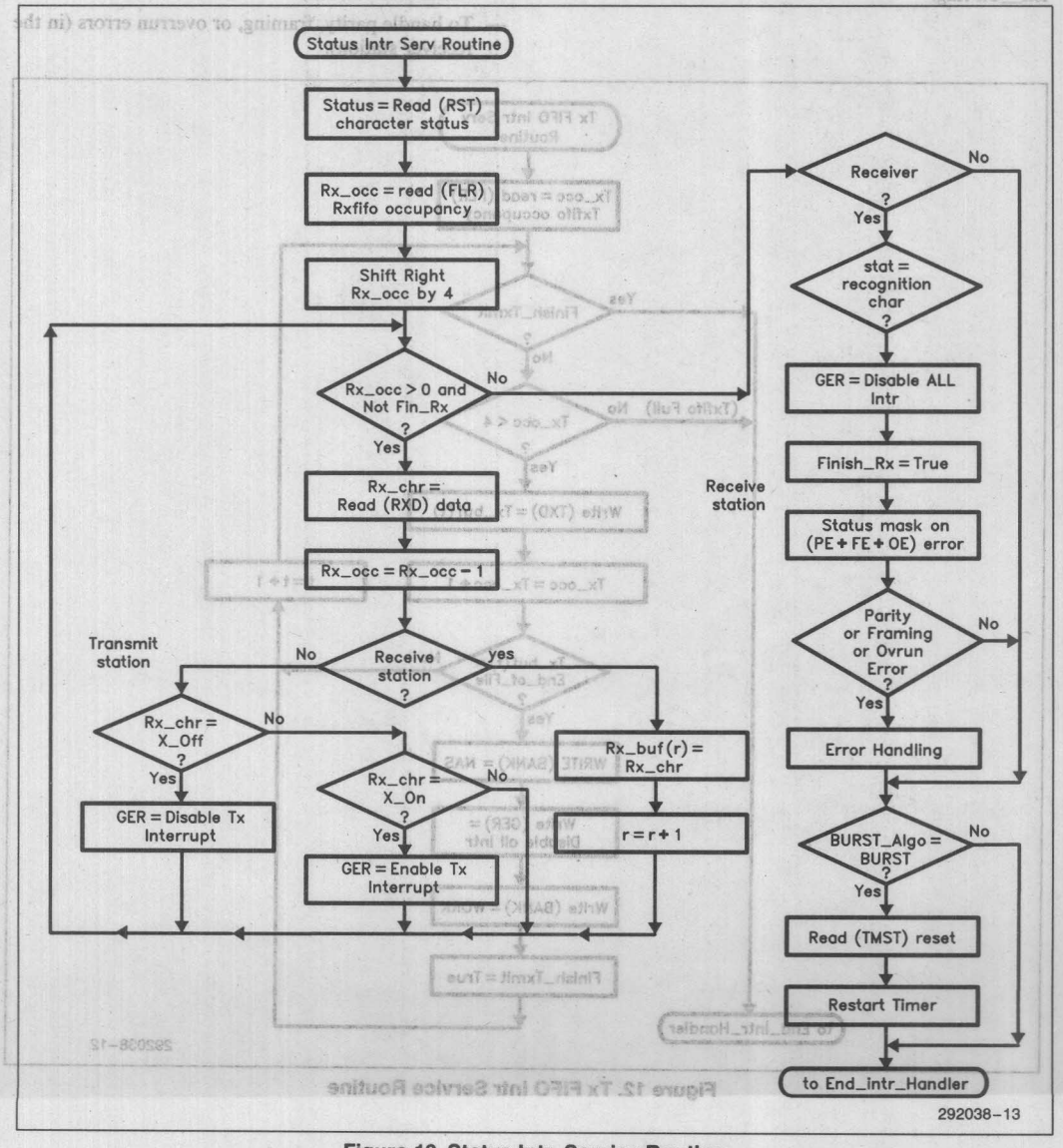


Figure 13. Status Intr. Service Routine

6.4.5 TIMER INTERRUPT SERVICE ROUTINE

A timer interrupt occurs when the receive character rate becomes low. The timer interrupt service routine first empties the Rx FIFO and then switches the burst algorithm to HUNTING mode.

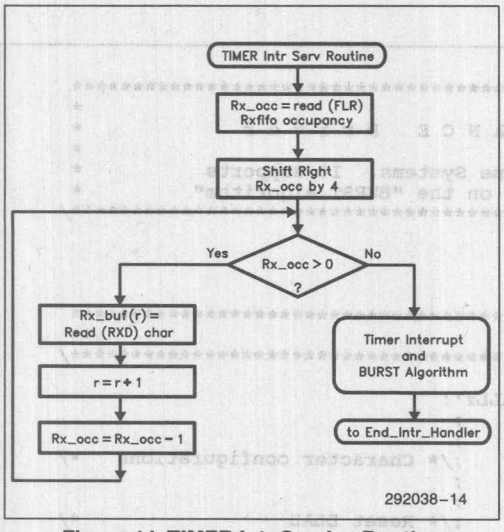


Figure 14. TIMER Intr Service Routine

6.4.6 MODEM INTERRUPT SERVICE ROUTINE

Modem interrupt occurs if one of the modem lines has dropped during transmission or reception. The modem interrupt service routine reads the MSR register to acknowledge the modem interrupt. The modem error routine is then executed.

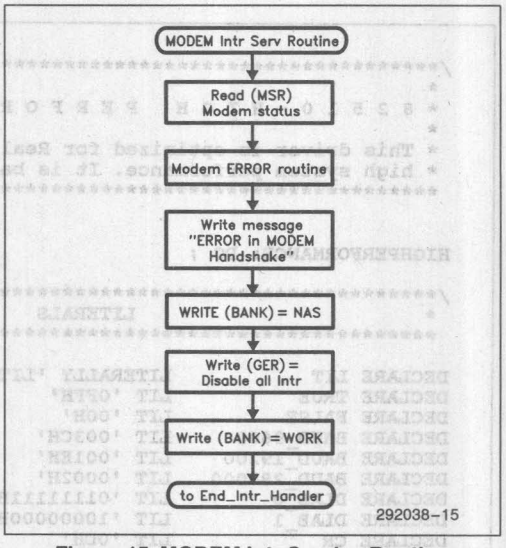


Figure 15. MODEM Intr Service Routine

APPENDIX A
PL/M SOURCE FILE

```
/*************************************************************************
 *                                                                       *
 * 8 2 5 1 0 - H I G H   P E R F O R M A N C E   D r i v e r            *
 *                                                                       *
 * This driver is optimized for Real Time Systems.  It supports          *
 * high system performance. It is based on the "BURST algorithm"         *
 *************************************************************************/

HIGHPERFORMANCE: DO ;

/*************************************************************************
 *                        LITERALS                                       *
 *************************************************************************/

DECLARE LIT             LITERALLY 'LITERALLY';
DECLARE TRUE            LIT '0FFH'          ;
DECLARE FALSE           LIT '00H'           ;
DECLARE BAUD_9600       LIT '003CH'         ;/* Character configurations   */
DECLARE BAUD_19200      LIT '001EH'         ;
DECLARE BAUD_288000     LIT '0002H'         ;
DECLARE DLAB_0          LIT '01111111B'     ;/* Reset DLAB                 */
DECLARE DLAB_1          LIT '10000000B'     ;/* Set   DLAB                 */
DECLARE CR              LIT '0DH'           ;/* Control characters         */
DECLARE LF              LIT '0AH'           ;
DECLARE X_Off           LIT '13H'           ;
DECLARE X_On            LIT '11H'           ;
DECLARE End_Of_File     LIT '1AH'           ;
DECLARE BASE_510        LIT '080H'          ;/* 8 2 5 1 0   registers      */
DECLARE NAS0            LIT '00000000B'     ;
DECLARE WORK1           LIT '00100000B'     ;
DECLARE GEN2            LIT '01000000B'     ;
DECLARE MODM3           LIT '01100000B'     ;
DECLARE TXD             LIT 'BASE_510 + 0'  ;/* BANK 0 - NAS               */
DECLARE RXD             LIT 'BASE_510 + 0'  ;
DECLARE BAL             LIT 'BASE_510 + 0'  ;
DECLARE BAH             LIT 'BASE_510 + 2'  ;
DECLARE GER             LIT 'BASE_510 + 2'  ;
DECLARE GIR             LIT 'BASE_510 + 4'  ;
DECLARE BANK            LIT 'BASE_510 + 4'  ;
DECLARE LCR             LIT 'BASE_510 + 6'  ;
DECLARE MCR             LIT 'BASE_510 + 8'  ;
DECLARE LSR             LIT 'BASE_510 +10'  ;
DECLARE MSR             LIT 'BASE_510 +12'  ;
DECLARE ACR0            LIT 'BASE_510 +14'  ;
DECLARE RXF             LIT 'BASE_510 + 2'  ;/* BANK 1 - WORK              */
DECLARE TXF             LIT 'BASE_510 + 2'  ;
DECLARE TMST            LIT 'BASE_510 + 6'  ;
DECLARE TMCR            LIT 'BASE_510 + 6'  ;
DECLARE FLR             LIT 'BASE_510 + 8'  ;
DECLARE RST             LIT 'BASE_510 +10'  ;
DECLARE RCM             LIT 'BASE_510 +10'  ;
DECLARE TCM             LIT 'BASE_510 +12'  ;
DECLARE GSR             LIT 'BASE_510 +14'  ;
DECLARE ICM             LIT 'BASE_510 +14'  ;
DECLARE FMD             LIT 'BASE_510 + 2'  ;/* BANK 2 - GENERAL CONFIGURE */
DECLARE TMD             LIT 'BASE_510 + 6'  ;
```

292038-16

```
DECLARE IMD             LIT 'BASE_510 + 8'  ;
DECLARE ACR1            LIT 'BASE_510 +10'  ;
DECLARE RIE             LIT 'BASE_510 +12'  ;
DECLARE RMD             LIT 'BASE_510 +14'  ;
DECLARE CLCF            LIT 'BASE_510 + 0'  ;/* BANK 3 - MODEM          */
DECLARE BBL             LIT 'BASE_510 + 0'  ;/* DLAB=1                  */
DECLARE BACF            LIT 'BASE_510 + 2'  ;
DECLARE BBH             LIT 'BASE_510 + 2'  ;/* DLAB=1                  */
DECLARE BBCF            LIT 'BASE_510 + 6'  ;
DECLARE PMD             LIT 'BASE_510 + 8'  ;
DECLARE MIE             LIT 'BASE_510 +10'  ;
DECLARE TMIE            LIT 'BASE_510 +12'  ;
DECLARE OUT2_MCR        LIT '00001000B'     ;/* Specific register bits  */
DECLARE DTR_MCR         LIT '00000001B'     ;
DECLARE DSR_MSR         LIT '00100000B'     ;
DECLARE CLRSTAT_ICM     LIT '00000100B'     ;
DECLARE INTR_510        LIT '21H'           ;
DECLARE PORT_80130M     LIT '0E2H'          ;
DECLARE EN_80130        LIT '0FDH'          ;
DECLARE PORT_EOI        LIT '0E0H'          ;
DECLARE COMM_EOI        LIT '61H'           ;/* End Of Interrupt command */
DECLARE ENRTX_GER       LIT '00001111B'     ;/* Enable Interrupt bits   */
DECLARE ENTX_GER        LIT '00000010B'     ;
DECLARE ENTXSTAT_GER    LIT '00001110B'     ;
DECLARE ENRX_GER        LIT '00001101B'     ;
DECLARE ENTIMRX_GER     LIT '00101101B'     ;
DECLARE DISTX_GER       LIT '00001101B'     ;
DECLARE DISRX_GER       LIT '00000010B'     ;/* Disable Interrupt bits  */
DECLARE DISRTX_GER      LIT '00000000B'     ;
DECLARE TXTHRESH0_FMD   LIT '00000000B'     ;/* FIFO threshold          */
DECLARE RXTHRESH0_FMD   LIT '00000000B'     ;
DECLARE RXTHRESH3_FMD   LIT '00110000B'     ;
DECLARE MASK_RXOCC      LIT '01110000B'     ;/* Mask on occupancy bits  */
DECLARE MASK_TXOCC      LIT '00000111B'     ;
DECLARE MASK_ACRSTAT    LIT '01000000B'     ;/* Mask on ACR status bits */
DECLARE CHRLEN_8        LIT '00000011B'     ;/* Async parameters        */
DECLARE STPBIT_1        LIT '00000000B'     ;
DECLARE PARITY_NON      LIT '00000000B'     ;
DECLARE SWRES_CMND      LIT '00010000B'     ;
DECLARE ERRCHR_RST      LIT '00001110B'     ;
DECLARE ACRSTAT_RIE     LIT '01000000B'     ;
DECLARE ACRSTAT_RST     LIT '01000000B'     ;
DECLARE NONI_GIR        LIT '00100001B'     ;/* Interrupt vector        */
DECLARE MODMI_GIR       LIT '00100000B'     ;
DECLARE   TXI_GIR       LIT '00100010B'     ;
DECLARE   RXI_GIR       LIT '00100100B'     ;
DECLARE STATI_GIR       LIT '00100110B'     ;
DECLARE  TIMI_GIR       LIT '00101010B'     ;
DECLARE AUTOACK_IMD     LIT '00001000B'     ;
DECLARE TIMOD_BBCF      LIT '00000000B'     ;/* Timer                   */
DECLARE TIMBI_TMIE      LIT '00000010B'     ;
DECLARE FIFO_IMD        LIT '00000000B'     ;
DECLARE STARTIMB_TMCR   LIT '00100010B'     ;
DECLARE STARTIMB_TMST   LIT '00000010B'     ;
DECLARE RTXCLK_BRGA_CLCF LIT '01010000B'    ;
DECLARE LOW_BAUD        LIT '00H'           ;/* BURST algorithm         */
DECLARE HUNTING_MODE    LIT '01H'           ;
DECLARE  SINGLE_MODE    LIT '02H'           ;
DECLARE   BURST_MODE    LIT '03H'           ;
DECLARE TIME_EXP        LIT '0FFFFH'        ;/* timeout=7mS (at 18.4 Mhz) */
DECLARE WAIT_TIME       LIT '00FFFH'        ;/* WAIT_FOR_MODEM_STATUS   */
```

292038-17

2

```
/*************************************************************************
 *                          VARIABLES                                    *
 *************************************************************************/

DECLARE TX_PTR POINTER PUBLIC   ;          /* Transmit buffer           */
DECLARE TX_BUF BASED TX_PTR (3000) BYTE ;
DECLARE IX_TX       WORD PUBLIC ;
DECLARE RX_BUF(3000) BYTE PUBLIC;          /* Receive  buffer           */
DECLARE IX_RX       WORD PUBLIC ;
DECLARE INTR_VEC    BYTE PUBLIC ;
DECLARE FIN_TX      BYTE PUBLIC ;          /* Finish Transmission flag  */
DECLARE FIN_RX      BYTE PUBLIC ;          /* Finish Reception    flag  */
DECLARE RX_CHR      BYTE PUBLIC ;
DECLARE TX_CHR      BYTE PUBLIC ;
DECLARE TX_OCC      BYTE PUBLIC ;
DECLARE RX_OCC      BYTE PUBLIC ;
DECLARE STAT        BYTE PUBLIC ;
DECLARE BAUD        WORD PUBLIC ;
DECLARE TEMP        BYTE PUBLIC ;
DECLARE FIN         BYTE PUBLIC ;
DECLARE SELECTION   BYTE PUBLIC ;
DECLARE RECEIVER    BYTE PUBLIC ;          /* Receive station           */
DECLARE BURST_ALGO  BYTE PUBLIC ;          /* BURST algorithm           */
DECLARE MODEM_HANDSHAKE BYTE PUBLIC ;
DECLARE COUNTER     WORD PUBLIC ;
DECLARE RX_ERROR    BYTE PUBLIC ;          /* Error occurred during     */
                                           /* reception                 */

/*-----------------------------------------------------------------------*/

/* I/O console utilities                                                 */
$INCLUDE (:F1:TIOHP.PEX)

/* Setup and H/W configurations                                          */
$INCLUDE (:F1:HPUTIL.PEX)

DECLARE MAIN LABEL PUBLIC ;

/*************************************************************************
 * Procedure   INITIALIZATIONS                                           *
 *************************************************************************
 * input:      none                                                      *
 * output:     none                                                      *
 * function:   driver initialization: parameters, 82510                  *
 *             configuration, modem status check.                        *
 * called by:  Main                                                      *
 * calling:    CONFIG_82510, INITIALIZE_BURST, WAIT_FOR_MODEM            *
 *                                                                       *
 * Init the Interrupt mechanism  by  enable Interrupt in GER register    *
 * At the Receive station: Enable Rx FIFO, Status and Modem Interrupts    *
 *                         Disable Timer Interrupt                       *
 * At the Transmit station: Enable Tx FIFO, Status and Modem Interrupts   *
 *                                                                       *
 * flowchart:  figure 7      description:  paragraph 6.3                 *
 *************************************************************************/

INITIALIZATIONS: PROCEDURE PUBLIC ;

   DISABLE ;
   CALL SET$INTERRUPT(INTR_510,INTR_HANDLER) ;
                                 /* Install THE INTR HANDLER            */

   TX_CHR=00 ;                   /* Clear TX_CHR and RX_CHR             */
   RX_CHR=00 ;
```

292038-18

```
     CALL TEXT ;                  /* TX_PTR is a pointer to the transmitted*/
                                  /* data                                  */
     IX_TX= 0FFFFH ;              /* The index buffer are assigned to -1   */
     IX_RX= 0FFFFH ;
     FIN_TX=FALSE ;               /* Init Finish Transmit and receive flags*/
     FIN_RX=FALSE ;
     RX_BUF(0)=0 ;
     RX_ERROR=FALSE ;             /* Reset the flag                        */

     BAUD=BAUD_288000 ;           /* The Async communication Baud rate is  */
                                  /* the 82510-full scale 288000           */

     CALL CONFIG_82510 ;          /* Configured the 82510:                 */
                                  /* S/W reset, character length, parity,  */
                                  /* stop bit, baud rate and fifo threshol */

/*************************************************************************
 *              INITIALIZE_BURST                                         *
 *************************************************************************
 * input:       none                                                     *
 * output:      Burst_Algo                                               *
 * function:    start Burst algorithm in Hunting mode                    *
 * called by:   INITIALIZATIONS                                          *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart:   figure  4          description:  paragraph 6.2.1         *
 *************************************************************************/

     IF BAUD<=BAUD_9600    THEN BURST_ALGO=HUNTING_MODE ;
                                  /* HUNTING mode:                         */
                                  /* Rx FIFO threshold is 0                */
                                  /* Timer interrupt is disable            */
     ELSE  BURST_ALGO=LOW_BAUD ;

     CALL WAIT_FOR_MODEM_STATUS ;
                                  /* Wait for Modem handshake line "DSR"   */
                                  /* if ACTIVE set     MODEM_HANDSHAKE     */
     TEMP = INPUT(RXD) ;
     TEMP = INPUT(RXD) ;
     TEMP = INPUT(RXD) ;
     TEMP = INPUT(RST) ;

END INITIALIZATIONS ;

/*************************************************************************
 * Procedure    CONFIG_82510                                             *
 *************************************************************************
 * input:       none                                                     *
 * output:      none                                                     *
 * function:    configure the 82510 to a specific operation              *
 *              mode                                                     *
 * called by:   INITIALIZATIONS                                          *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart:   figure 8         description:  paragraph 6.3             *
 *************************************************************************/

CONFIG_82510: PROCEDURE PUBLIC ;

/* Perform Software reset                                               */
OUTPUT(BANK) = WORK1;             /* Move to work bank                  */
OUTPUT(ICM) = SWRES_CMND;         /* S/W reset command                  */
```

292038-19

```
/* BANK  ZERO -  NAS  (The default BANK)                          */

/* Configured the character by writing to LCR:                    */
/* 1 stop bit, 8 bit lengh, non parity                            */
OUTPUT(LCR)=(STPBIT_1  + CHRLEN_8 + PARITY_NON) ;
OUTPUT(MCR)=(DTR_MCR OR OUT2_MCR) ;
                                /* Required only in IBM PC environment:  */
                                /* set OUT2 signal to control an external*/
                                /* 3-state buffer that drives the 82510  */
                                /* interrupt signal                      */

IF RECEIVER THEN OUTPUT(ACR0)=End_Of_File ;
                                /* At the Receive station EOF is         */
                                /* recognized to terminate reception     */
ELSE OUTPUT(ACR0)= X_OFF  ;     /* At the Transmit station "X Off" is    */
                                /* recognized to stop transmission       */
                                /* temporary                             */

                                /* Enable 82510 Interrupt by set GER,    */
                                /* done at the end of INITIALIZATIONS    */

                                /* Init the 82510 Interrupt mechanism    */
DISABLE ;
IF RECEIVER THEN OUTPUT(GER)=ENRX_GER ;
                                /* at the  Receive  station              */
ELSE OUTPUT(GER)=ENTXSTAT_GER  ;
                                /* and the Transmit station              */

                                /* Configured baud rate to 288000        */
                                /* by writing to BRG A (BAL and BAH)     */
OUTPUT(LCR)=INPUT(LCR) OR DLAB_1;  /*Set DLAB to allow access to BRG     */
OUTPUT(BAL)=LOW (BAUD_288000) ;
OUTPUT(BAH)=HIGH(BAUD_288000) ;
OUTPUT(LCR)=INPUT(LCR) AND DLAB_0; /* reset DLAB                       */

/* BANK TWO - General configuration                               */
OUTPUT(BANK)=GEN2 ;

OUTPUT(IMD)=(AUTOACK_IMD OR FIFO_IMD) ;
                                /* Automatic interrupt acknowledge,      */
                                /* Rxfifo depth is four bytes            */

OUTPUT(FMD)=(TXTHRESH0_FMD OR RXTHRESH0_FMD) ;
                                /* Rxfifo threshold is temporally zero   */
                                /* for HUNTING mode (BURST algorithm)    */
                                /* Txfifo threshold is zero for max      */
                                /* interrupt latency                     */

IF RECEIVER THEN OUTPUT(ACR1)=End_Of_File ;
                                /* At the Receive station EOF is         */
                                /* recognized, the same as ACR0          */
ELSE OUTPUT(ACR1)=X_ON ;        /* At the Transmit station "X On"  is    */
                                /* recognized to continue transmission   */

OUTPUT(RIE) = (ACRSTAT_RIE OR INPUT(RIE)) ;
                                /* Enable interrupt on programmed control*/
                                /* character received (ACR0/ACR1)        */

/* BANK THREE - MODEM configuration                               */
OUTPUT(BANK)=MODM3 ;

OUTPUT(BBCF)=(TIMOD_BBCF) ;     /* BRG B configured to TIMER mode        */
OUTPUT(BANK) = NAS0;            /* Move to nas bank to set DLAB          */
OUTPUT(LCR)=INPUT(LCR) OR DLAB_1 ; /* Set DLAB to allow access to BRG    */
OUTPUT(BANK) = MODM3;           /* MODEM bank                            */
OUTPUT(BBL) = LOW (TIME_EXP);   /* Set max timeout (7ms if 18Mhz crystal)*/
```

292038-20

```
OUTPUT(BBH) = HIGH(TIME_EXP);    /* to issue interrupt when time has      */
OUTPUT(BANK) = NAS0;             /* expired. Move to NAS bank again       */
OUTPUT(LCR) =INPUT(LCR) AND DLAB_0 ; /* Reset DLAB                        */
OUTPUT(BANK) = MODM3;            /* Switch to BANK THREE - MODEM          */
OUTPUT(CLCF)=RTXCLK_BRGA_CLCF ; /* The receive and transmit clock source */
                                 /* is BRG A                              */

OUTPUT(TMIE)=TIMBI_TMIE ;        /* Enable Timer block interrupt          */
                                 /* (stil disabled in Timer bit in GER)   */

/* BANK ONE - general WORK       - The RUNTIME bank                       */
OUTPUT(BANK)=WORK1 ;
OUTPUT(ICM)=CLRSTAT_ICM ;        /* Issues a command to clear all         */
                                 /* status registers                      */


/* Remain in   W O R K - THE runtime  bank                                */


END CONFIG_82510 ;

/************************************************************************
 * Procedure   WAIT_FOR_MODEM_STATUS                                    *
 ************************************************************************
 * input:      none                                                     *
 * output:     Modem_Handshake                                          *
 * function:   waits with a timeout for DSR active,                     *
 *             returns status flag                                      *
 * called by:  INITIALIZATIONS                                          *
 * calling:    none                                                     *
 *                                                                      *
 * flowchart:  figure 9        description:  paragraph 6.3.1            *
 ************************************************************************/

WAIT_FOR_MODEM_STATUS: PROCEDURE PUBLIC ;

MODEM_HANDSHAKE = FALSE ;
COUNTER = WAIT_TIME ;

DO WHILE (NOT MODEM_HANDSHAKE) AND ((COUNTER:=COUNTER-1) > 0 ) ;
   IF (INPUT(MSR) AND DSR_MSR) <> 0   THEN MODEM_HANDSHAKE = TRUE ;
END ;

END WAIT_FOR_MODEM_STATUS ;

/************************************************************************
 * Procedure   INTERRUPT HANDLER                                        *
 ************************************************************************
 * input:      Tx Buffer                                                *
 * output:     Rx Buffer, Finish_Tx, Finish_Rx                          *
 * function:   service all 82510 interrupt sources:                     *
 *             Rx Fifo, Tx Fifo, Status, Timer, Modem                   *
 * called by:  82510 hardware interrupt                                 *
 * calling:    Rx_Fifo_Intr, Tx_Fifo_Intr, Status_Intr,                 *
 *             Timer_Intr, Modem_Intr                                   *
 *                                                                      *
 * flowchart:  figure 10   description:  paragraph 6.4, 6.4.1           *
 ************************************************************************/

INTR_HANDLER: PROCEDURE INTERRUPT INTR_510 REENTRANT PUBLIC ;

ENABLE  ;                        /* Enable Interrupts of                  */
                                 /* HIGHIER priority devices              */

INTR_VEC=INPUT(GIR);             /* Get the 82510-highest priority        */
                                 /* pending interrupt                     */
```

292038-21

```
/*************************************************************************
 *          Rx_FIFO_INTR                                                 *
 *************************************************************************
 * input:       none                                                     *
 * output:      Rx_Buffer, Burst_Algo                                    *
 * function:    service Rx Fifo interrupt                                *
 *              receive characters; store in receive buffer              *
 * called by:   INTERRUPT HANDLER                                        *
 * calling:     BURST_ALGO                                               *
 *                                                                       *
 * flowchart:   figure 11      description:  paragraph 6.4.2             *
 *************************************************************************/

IF INTR_VEC=RXI_GIR THEN DO ;

   RX_OCC=INPUT(FLR) ;              /* Rx fifo level occupancy          */
                                    /* Shift the Rx occupancy bit       */
   RX_OCC=SHR(RX_OCC,4) ;           /* to get it's real value           */
                                    /* - OPTIMIZE code -                */
                                    /* Empty the Rx FIFO and store the  */
                                    /* received character in RX_BUF     */
   RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ;
                                    /* Read the first character immediatly */
                                    /* to save Real Time                */
   DO WHILE (RX_OCC:=RX_OCC-1) > 0 ;
      RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ;
   END ;



/*************************************************************************
 *          BURST_ALGORITHM                                              *
 *************************************************************************
 * input:       Burst_Algo                                               *
 * output:      Burst_Algo                                               *
 * function:    execute a step in the burst algorithm                    *
 *              after characters are received                            *
 * called by:   Rx_FIFO_INTR                                             *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart: figure 5   description: par. 6.2.2.1 to 6.2.2.3           *
 *************************************************************************/

      /*-----------------------------------------------------------------*
       *  B U R S T     M O D E - step 3  (full fifo threshold)          *
       * Reset the Timer status                                          *
       * Restart the Timer                                               *
       *-----------------------------------------------------------------*/
      IF BURST_ALGO = BURST_MODE THEN DO ;
         TEMP = INPUT(TMST);
         OUTPUT(TMCR) =STARTIMB_TMCR;
      END;


      /*-----------------------------------------------------------------*
       *   H U N T I N G   M O D E - step 1                              *
       * Operate the TIMER                                               *
       * Change to step 2 SINGLE  mode                                   *
       *-----------------------------------------------------------------*/
      ELSE IF BURST_ALGO = HUNTING_MODE THEN DO ;
         OUTPUT(TMCR)=STARTIMB_TMCR ;
         BURST_ALGO=SINGLE_MODE ;
      END ;
```

292038-22

```
      /*-------------------------------------------------------------------*
       *   S I N G L E     M O D E - step 2                                 *
       * If TIME has expired, means the receive                            *
       * rate is LOW, return to  HUNTING mode                              *
       * If TIME did NOT expire, means the                                 *
       * Receive rate is HIGH, set Rx FIFO threshold, Restart the          *
       * Timer and switch to BURST mode                                    *
       *-------------------------------------------------------------------*/
      ELSE IF BURST_ALGO = SINGLE_MODE THEN DO ;

        IF ((INPUT(TMST) AND STARTIMB_TMST) <>0) THEN
                    BURST_ALGO= HUNTING_MODE ;
        ELSE DO;
           OUTPUT(BANK) = GEN2;/* Switch to BANK TWO - General Config   */
           OUTPUT(FMD)=TXTHRESH0_FMD OR RXTHRESH3_FMD;
           OUTPUT(BANK) =NAS0; /* Switch to BANK ZERO - NAS             */
           OUTPUT(GER) = ENTIMRX_GER;
                               /* Enable TIMER,RX and  MODEM interrupts */
           OUTPUT(BANK)=WORK1; /* Switch to BANK ONE  - WORK            */
           BURST_ALGO = BURST_MODE;
           TEMP = INPUT(TMST); /* Reset timer status                    */
           OUTPUT(TMCR) = STARTIMB_TMCR;
        END;
      END;                     /* End of SINGLE mode                    */

    /* ....End of BURST algorithm.....................................*/

    /*  Another try to empty the Rx fifo                               */
    /*  before leaving the interrupt handler                           */

      DO WHILE (INPUT(FLR)<>0)  ;
                               /* Empty the Rx FIFO and store the       */
                               /* received character in RX_BUF          */
        RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ;
      END ;

  END ;                        /* End of Rx fifo interrupt              */

  /*********************************************************************
   *          TxFIFO_INTR                                              *
   *********************************************************************
   * input:       Tx_Buffer                                            *
   * output:      Finish_tx                                            *
   * function:    service Tx Fifo interrupt                            *
   *              transmit characters from transmit buffer (OPTIMIZE code)  *
   * called by:   INTERRUPT HANDLER                                    *
   * calling:     none                                                 *
   *                                                                   *
   * flowchart:   figure 12        description:  paragraph 6.4.3       *
   *********************************************************************/

  ELSE IF INTR_VEC=TXI_GIR THEN DO ;
     TX_OCC=INPUT(FLR) AND MASK_TXOCC ;
                               /* Tx fifo level occupancy               */
      /* Fill Tx FIFO, the transmitted characters are taken from TX_buf */
     DO WHILE (TX_OCC:=TX_OCC+1)<5 ;
        OUTPUT(TXD)=TX_BUF(IX_TX:=IX_TX+1);
        IF TX_BUF(IX_TX)=End_Of_File THEN DO ;
           OUTPUT(BANK)=NAS0 ;    /* Disable Tx interrupt, as the transmit */
           OUTPUT(GER)=DISTX_GER; /* delimiter character  was identified   */
           OUTPUT(BANK)=WORK1 ;   /* Switch to BANK ONE  - WORK            */
           TX_OCC = 5 ;           /* load TX_OCC to terminate external loop*/
           FIN_TX = TRUE ;        /*  Set Finish transmit flag             */
        END ;
     END ;
  END ;                        /* End of TXFIFO_INTR                    */
```

292038-23

```
/*************************************************************************
 *              STATUS_INTR                                              *
 *************************************************************************
 * input:       none                                                     *
 * output:      Finish_Rx                                                *
 * function:    service Status interrupt                                 *
 *              Receive  station: EOF terminate the reception            *
 *              Transmit station: X_Off Disable the transmission         *
 *                                X_On  Enable  the transmission         *
 * called by:   INTERRUPT HANDLER                                        *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart:   figure 13       description:  paragraph 6.4.4            *
 *************************************************************************/

ELSE IF INTR_VEC=STATI_GIR THEN DO ;

    STAT=INPUT(RST) ;                /* Get the current RST status       */

    RX_OCC=INPUT(FLR) ;              /* Rx fifo level occupancy          */
    RX_OCC=SHR(RX_OCC,4) ;

    DO WHILE (RX_OCC>0 AND (NOT FIN_RX));
       RX_OCC=RX_OCC-1 ;             /*         First, empty Rx FIFO     */
       RX_CHR=INPUT(RXD) ;

       IF RECEIVER THEN   RX_BUF(IX_RX:=IX_RX+1)=RX_CHR ;

       ELSE  DO ;
          IF RX_CHR = X_OFF THEN DO ;
             OUTPUT(BANK)=NAS0;/* Switch to BANK ZERO - NAS              */
             OUTPUT(GER) = INPUT(GER) AND DISTX_GER ;
                              /* Disable Transmit interrupt               */
             OUTPUT(BANK)=WORK1;/* Switch to BANK ONE  - WORK             */
          END ;
          ELSE IF RX_CHR = X_ON THEN DO ;
             OUTPUT(BANK)= NAS0 ;
             OUTPUT(GER) = INPUT(GER) OR  ENTX_GER ;
                              /* Enable  Transmit interrupt  again        */
             OUTPUT(BANK)= WORK1 ;
          END ;
       END ;
    END ;

  IF RECEIVER THEN DO ;
     IF ((STAT AND ACRSTAT_RST) <> 0) THEN DO ;
        OUTPUT(BANK)= NAS0 ;   /* If End_Of_Line was recognized,           */
        OUTPUT(GER) = DISRTX_GER ;
        OUTPUT(BANK)= WORK1 ;  /* Disable 82510-interrupts and the         */
        FIN_RX      = TRUE ;   /* Reception                                */
     END ;
     ELSE IF ((STAT AND ERRCHR_RST) <> 0) THEN DO ;
        CALL WRITE(@('** ERROR in character Status ',0)) ;
        CALL ERROR_CHAR_HANDLER ;

        IF BURST_ALGO=BURST_MODE THEN DO ;
                              /* In BURST mode do:                        */
        TEMP = INPUT(TMST); /* Reset timer status                         */
        OUTPUT(TMCR) = STARTIMB_TMCR;
        END;                  /* Restart TIMER                            */
     END ;
   END ;

END ;                         /* End of STATUS interrupt                  */
```

292038-24

```
/*************************************************************************
 *              TIMER_INTR                                               *
 *************************************************************************
 * input:       none                                                     *
 * output:      Burst_Algo                                               *
 * function:    service Timer interrupt; receive characters              *
 *              and switch Burst_Algo to HUNTING mode                    *
 * called by:   INTERRUPT HANDLER                                        *
 * calling:     BURST &TIMER                                             *
 *                                                                       *
 * flowchart:   figure 14       description:  paragraph 6.4.5            *
 *************************************************************************/

ELSE IF INTR_VEC=TIMI_GIR THEN DO ;

      IF ((RX_OCC:=INPUT(FLR))<>0) THEN DO ;

         RX_OCC=SHR(RX_OCC,4) ; /* Rx fifo level occupancy, shift right  */
                                /* - OPTIMIZE code -                     */
                                /* Empty the Rx FIFO and store the       */
                                /* received character in RX_BUF          */
         RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ;
         DO WHILE (RX_OCC:=RX_OCC-1) > 0 ;
            RX_BUF(IX_RX:=IX_RX+1)=INPUT(RXD) ;
         END ;
                                /* Store the received character in RX_buf*/
      END ;



/*************************************************************************
 *              BURST & TIMER                                            *
 *************************************************************************
 * input:       Burst_Algo                                               *
 * output:      Burst_Algo                                               *
 * function:    execute a step in the burst algorithm                    *
 *              after timer interrupt; switch to HUNTING                 *
 * called by:   TIMER_INTR                                               *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart:   figure  6       description:  paragraph 6.2.2.4          *
 *************************************************************************/
   OUTPUT(BANK) = GEN2;          /* Switch to BANK TWO - General Config   */
   OUTPUT(FMD) = TXTHRESH0_FMD OR RXTHRESH0_FMD;
                                 /* Rxfifo threshold=0, Txfifo threshold=0*/

   OUTPUT(BANK) = NAS0;          /* Switch to BANK ZERO - NAS             */
   OUTPUT(GER) = ENRX_GER;       /* Disable Timer interrupt and           */
   OUTPUT(BANK) = WORK1;         /* Enable RX,STAT,MODEM interrupts       */
   TEMP = INPUT(TMST);           /* Acknowledge TIMER interrupt           */
   BURST_ALGO = HUNTING_MODE ;   /* Back to HUNTING mode                  */
END ;                            /* End of TIMER interrupt                */
```

292038-25

```
/*************************************************************************
 *              MODEM_INTR                                               *
 *************************************************************************
 * input:       none                                                     *
 * output:      none                                                     *
 * function:    service Modem interrupt and handle modem errors.         *
 *              Modem interrupt is occurred if No Modem was setup, or    *
 *              if DSR was dropped in the middle of the communication    *
 * called by:   INTERRUPT HANDLER                                        *
 * calling:     none                                                     *
 *                                                                       *
 * flowchart:   figure 15       description:  paragraph 6.4.6            *
 *************************************************************************/

ELSE IF INTR_VEC=MODMI_GIR THEN DO ;

   STAT=INPUT(MSR) ;              /* Get MODEM status                    */

   CALL ERROR_MODEM_HANDLER ;     /* Handel Modem Errors handshake       */

END ;                             /* End of MODEM interrupt              */


OUTPUT(PORT_EOI)=COMM_EOI ;       /* Write End_Of_Interrupt command to the */
                                  /* PIC (8259A)                         */
END INTR_HANDLER ;


/*************************************************************************
 * Procedure    ERROR_MODEM_HANDLER                                      *
 *************************************************************************/

ERROR_MODEM_HANDLER: PROCEDURE PUBLIC ;

MODEM_HANDSHAKE = FALSE ;         /* Flag indicates that an Error occurred */
                                  /* in Modem                            */

END ERROR_MODEM_HANDLER ;


/*************************************************************************
 * Procedure    ERROR_CHAR_HANDLER                                       *
 *************************************************************************/

ERROR_CHAR_HANDLER: PROCEDURE PUBLIC ;

RX_ERROR      = TRUE  ;           /* Flag indicates that an Error occurred */
                                  /* during Reception                    */

OUTPUT(BANK) = NAS0 ;             /* Switch to BANK ZERO - NAS           */
OUTPUT(GER)  = DISRTX_GER ;       /* Disable all the 82510 Interrupts    */
OUTPUT(BANK) = WORK1 ;            /* Switch to BANK ONE  - WORK          */

END ERROR_CHAR_HANDLER ;
```

292038-26

```
/*************************************************************************
 *  Procedure       LOOP                                                 *
 *                                                                       *
 * LOOP procedure is executed until Transmission/Reception Finishes      *
 * or until the loop ends.                                               *
 *************************************************************************/

LOOP: PROCEDURE PUBLIC ;
DECLARE N WORD ;
DECLARE NUM WORD ;.
DECLARE MAXLOOP BYTE ;
MAXLOOP= 20 ;
NUM=0 ;
DO WHILE ( (NOT FIN_TX) AND (NOT FIN_RX) AND (NUM<MAXLOOP) ) ;
   NUM=NUM+1 ;                  /* Count the LOOP times                */
   CALL WRITELN(@('... Background Program  ...',0));
   ENABLE ;
   CALL TIME(5000) ;            /* Software delay                      */

END ;

IF FIN_TX THEN CALL WRITELN(@('T r a n s m i s s i o n  -  E N D E D ',0));
IF FIN_RX THEN CALL WRITELN(@('R e c e p t i o n  - E N D E D ',0));

OUTPUT(BANK)=NAS0 ;             /* If Communication is Not ended       */
OUTPUT(GER)=DISRTX_GER ;        /* successfully the Interrupts are     */
OUTPUT(BANK)=WORK1 ;            /* Disabled by MAIN                    */
IF  RECEIVER THEN DO ;          /* Display RX buffer                   */
   IF FIN_RX THEN DO ;
      CALL WRITELN(@('The Received Message:',0)) ;
      CALL DISPTEXT(@RX_BUF) ;
   END ;
   ELSE
      CALL WRITELN(@('** ERROR -THE Reception NOT ended successfully',0)) ;
END ;
ELSE IF (NOT FIN_TX) THEN       /* The Transimt station                */
    CALL WRITELN(@('** ERROR -THE Transmission NOT ended successfully',0));
END LOOP ;

/*************************************************************************
 *  Procedure       TEXT                                                 *
 *************************************************************************
 * input:      none                                                      *
 * output:     Tx_ptr                                                    *
 * function:   Return a pointr to the Transmit buffer.  Data in the      *
 *             transmit buffer must be trminated by End_Of_File.         *
 * called by:  INITIALIZATIONS                                           *
 * calling:    none                                                      *
 *************************************************************************/

TEXT: PROCEDURE PUBLIC ;

TX_PTR=@('>',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz0123456789',
CR,LF,End_Of_File,0) ;
                                /* End_Of_File-terminate the Transmission*/
END TEXT ;
```

292038-27

```
/*************************************************************************
 * External    procedures                                                *
 *************************************************************************
 * WRITELN:  I/O console utility - dispaly a string, end with CR         *
 * MENU:     I/O console utility - display a menu, enter the user        *
 *                                 selection                             *
 * DISPTEXT: I/O console utility - display the contents of the           *
 *                                 Receive buffer (Rx_buf)               *
 * INIT_HARDWARE_SETUP: Setup and Hardware configurations of the         *
 *                      specific station                                 *
 *************************************************************************/


/*************************************************************************
 * Procedure   MAIN                                                      *
 *************************************************************************
 * input:      Finish_Rx, Finish_Tx                                      *
 * output:     Receiver flag                                             *
 * function:   get station type (Rx or Tx) from the operator;            *
 *             wait till communication is completed; display;            *
 *               RECEIVER STATION SHOULD BE ACTIVATED FIRST              *
 * called by:  Application                                               *
 * calling:    INITIALIZATIONS, LOOP                                     *
 *                                                                       *
 * flowchart:  figure  3      description:  paragraph 6.1                *
 *************************************************************************/

MAIN:

CALL INIT_HARDWARE_SETUP          ;
                                  /* External, Setup and H/W configurations*/
FIN=FALSE ;
DO WHILE NOT(FIN) ;
    SELECTION=0 ;
    CALL WRITELN(@('---------------------------------------------- ',0));
    SELECTION=MENU(SELECTION,@('Station: (Quit/Transmitter/Receiver)',0)) ;
                                  /* Get operator selection.              */
                                  /* Receiver station should be activated  */
                                  /* prior to the transmitter station      */
    DO CASE SELECTION ;
                FIN=TRUE     ;    /* 0 - Quit of HIGH PERFORMANCE Driver   */
                DO ;              /* 1 - Transmit station                  */
                        RECEIVER=FALSE ;
                        CALL INITIALIZATIONS ;
                        CALL LOOP ;
                END ;
                DO ;              /* 2 - Receive station                   */
                        RECEIVER=TRUE  ;
                        CALL INITIALIZATIONS ;
                        CALL LOOP ;
                END ;
    END ;
END ;

CALL EXIT ;

END HIGHPERFORMANCE ;

/*************************************************************************/
```

292038-28

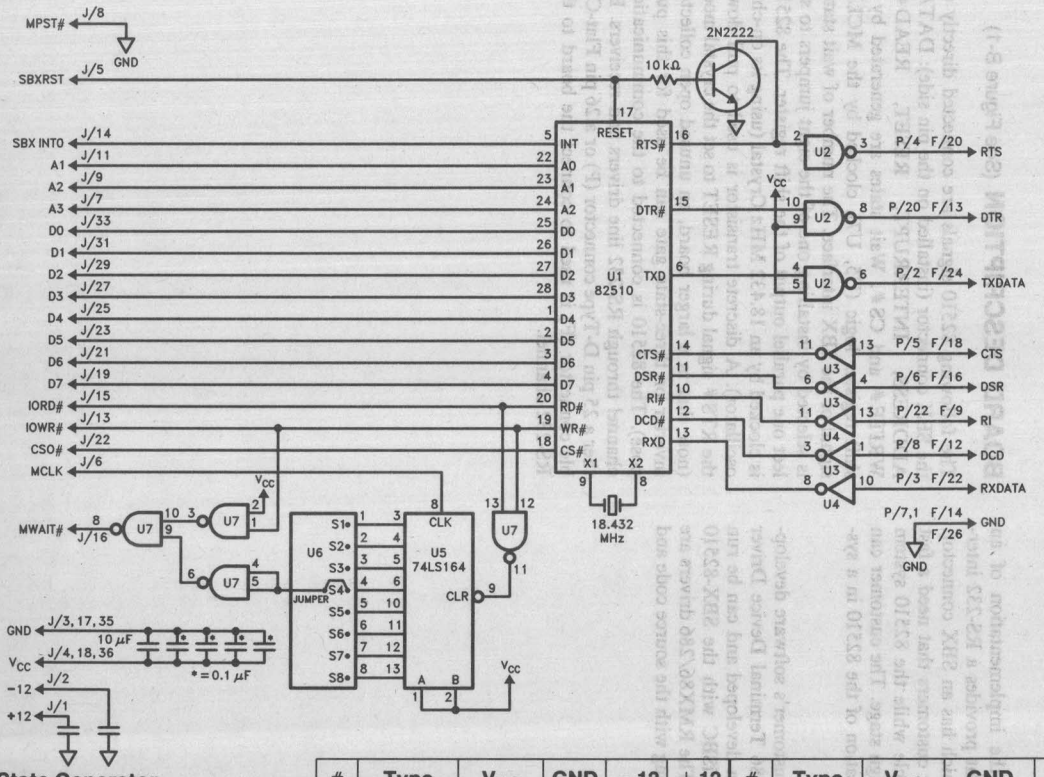
APPENDIX B
82510 BASED SBX SERIAL CHANNEL

This document describes the implementation of an 82510 based SBX board that provides a RS-232 interface to any iSBC board which has an SBX connector. The SBX can be useful for customers that need a fast software development vehicle while the 82510 system hardware is still in the design stage. The customer can also use the SBX for evaluation of the 82510 in a system environment.

In order to minimize the customer's software development costs, the RMX86/286 Terminal Device Driver for the 82510 has also been developed and can be run by the RMX user on his iSBC with the SBX-82510 board described herewith. The RMX86/286 drivers are available from INSITE, along with the source code and the documentation.

BOARD DESCRIPTION (See Figure B-1)

The following 82510 signals are connected directly to the SBX connector (installed on the pin side): DATA, ADDRESS, INTERRUPT, RESET, READ#, WRITE# and CS#. Wait states are generated by a shift register logic (U5, U7), clocked by the MCLK signal of the SBX interface. The number of wait states is selected by installing one of the eight jumpers to select one parallel output of the shift register. The 82510 is clocked by an 18.432 MHz Crystal (using its on-chip oscillator). A discrete transistor is used to pull down the RTS# signal during RESET to set the crystal mode (note that in a larger board, an unused open collector inverter or three-state gate can be used for this purpose). The 82510 is connected to the communication channel through RS-232 line drivers and receivers. Either a 25 pin D-Type connector (P) or a 26 pin Flat-Cable connector (F) is used to connect the board to the RS-232 channel.



WAIT State Generator

# of WAIT States	Jumper to CLOSE	# of WAIT States	Jumper to CLOSE
1	S1	5	S5
2	S2	6	S6
3	S3	7	S7
4	S4	8	S8

Only One Jumper Should Be Closed at a Time

#	Type	V_{CC}	GND	−12	+12	#	Type	V_{CC}	GND	−12	+12
U1	82510	21	7			U6	Jumper				
U2	1488	5, 9	7	1	14	U7	74LS00	14	7		
U3	1489	14	7			J	SBX Male Connector for 8 Bit Bus				
U4	1489	14	7			P	25 Pin D-Type Connector (Male)				
U5	74LS164	1, 2, 14	7			F	26 Pin Flat-cable Connector (Male)				

Either P *or* F should be installed.

Figure B-1. Board Schematics

November 1986

2

Using the 8273 SDLC/HDLC Protocol Controller

JOHN BEASTON
MICROCOMPUTER APPLICATIONS

Order Number: 611001-001

USING THE 8273 SDLC/ HDLC PROTOCOL CONTROLLER

CONTENTS

PAGE

INTRODUCTION 2-297

SDLC/HDLC OVERVIEW 2-297

BASIC 8273 OPERATION 2-300

HARDWARE ASPECTS OF THE 8273 2-301

CPU Interface 2-301

Modem Interface 2-305

SOFTWARE ASPECTS OF THE 8273 2-308

Command Phase Software 2-309

Execution Phase Software 2-310

Result Phase Software 2-311

8273 COMMAND DESCRIPTION 2-312

Initialization/Configuration Commands .. 2-313

Operating Mode Register 2-313

Serial I/O Mode Register 2-313

Data Transfer Mode Register 2-313

One Bit Delay Register 2-314

Receive Commands 2-314

General Receive 2-314

Selective Receive 2-314

Selective Loop Receive 2-314

Receive Disable 2-315

Transmit Commands 2-317

Transmit Frame 2-317

Loop Transmit 2-317

Transmit Transparent 2-317

Abort Commands 2-317

Reset Commands 2-318

Modem Control Commands 2-319

HDLC CONSIDERATIONS 2-319

LOOP CONFIGURATIONS 2-320

APPLICATION EXAMPLE 2-324

CONCLUSION 2-331

APPENDIX A 2-332

INTRODUCTION

The Intel 8273 is a Data Communications Protocol Controller designed for use in systems utilizing either SDLC or HDLC (Synchronous or High-Level Data Link Control) protocols. In addition to the usual features such as full duplex operation, automatic Frame Check Sequence generation and checking, automatic zero bit insertion and deletion, and TTL compatibility found on other single component SDLC controllers, the 8273 features a frame level command structure, a digital phase locked loop, SDLC loop operation, and diagnostics.

The frame level command structure is made possible by the 8273's unique internal dual processor architecture. A high-speed bit processor handles the serial data manipulations and character recognition. A byte processor implements the frame level commands. These dual processors allow the 8273 to control the necessary byte-by-byte operation of the data channel with a minimum of CPU (Central Processing Unit) intervention. For the user this means the CPU has time to take on additional tasks. The digital phase locked loop (DPLL) provides a means of clock recovery from the received data stream on-chip. This feature, along with the frame level commands, makes SDLC loop operation extremely simple and flexible. Diagnostics in the form of both data and clock loopback are available to simplify board debug and link testing. The 8273 is a dedicated function peripheral in the MCS-80/85 Microcomputer family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8273 as a component and shows its use in a generalized loop configuration and a typical 8085 system. The 8085 system was used to verify the SDLC operation of the 8273 on an actual IBM SDLC data communications link.

The first section of this application note presents an overview of the SDLC/HDLC protocols. It is fairly tutorial in nature and may be skipped by the more knowledgeable reader. The second section describes the 8273 from a functional standpoint with explanation of the block diagram. The software aspects of the 8273, including command examples, are discussed in the third section. The fourth and fifth sections discuss a loop SDLC configuration and the 8085 system respectively.

SDLC/HDLC OVERVIEW

SDLC is a protocol for managing the flow of information on a data communications link. In other words, SDLC can be thought of as an envelope—addressed, stamped, and containing an s.a.s.e.—in which information is transferred from location to location on a data communications link. (Please note that while SDLC is discussed specifically, all comments also apply to HDLC except where noted.) The link may be either point-to-point or multi-point, with the point-to-point configuration being either switched or nonswitched. The information flow may use either full or half duplex exchanges. With this many configurations supported, it is difficult to find a synchronous data communications application where SDLC would not be appropriate.

Aside from supporting a large number of configurations, SDLC offers the potential of a 2× increase in throughput over the presently most prevalent protocol: Bi-Sync. This performance increase is primarily due to two characteristics of SDLC: full duplex operation and the implied acknowledgement of transferred information. The performance increase due to full duplex operation is fairly obvious since, in SDLC, both stations can communicate simultaneously. Bi-Sync supports only half-duplex (two-way alternate) communication. The increase from implied acknowledgement arises from the fact that a station using SDLC may acknowledge previously received information while transmitting different information. Up to 7 messages may be outstanding before an acknowledgement is required. These messages may be acknowledged as a block rather than singly. In Bi-Sync, acknowledgements are unique messages that may not be included with messages containing information and each information message requires a separate acknowledgement. Thus the line efficiency of SDLC is superior to Bi-Sync. On a higher level, the potential of a 2× increase in performance means lower cost per unit of information transferred. Notice that the increase is not due to higher data link speeds (SDLC is actually speed independent), but simply through better line utilization.

Getting down to the more salient characteristics of SDLC; the basic unit of information on an SDLC link is that of the frame. The frame format is shown in Figure 1. Five fields comprise each frame: flag, address, control, information, and frame check sequence. The flag fields (F) form the boundary of the frame and all

Opening Flag	Address Field (A)	Control Field (C)	Information Field (I)	Frame Check Sequence (FCS)	Closing Flag
01111110	8 Bits	8 Bits	Any Length 0 to N Bits	16 Bits	01111110

Figure 1. SDLC Frame Format

other fields are positionally related to one of the two flags. All frames start with an opening flag and end with a closing flag. Flags are used for frame synchronization. They also may serve as time-fill characters between frames. (There are no intraframe time-fill characters in SDLC as there are in Bi-Sync.) The opening flag serves as a reference point for the address (A) and control (C) fields. The frame check sequence (FCS) is referenced from the closing flag. All flags have the binary configuration 01111110 (7EH).

SDLC is a bit-oriented protocol, that is, the receiving station must be able to recognize a flag (or any other special character) at any time, not just on an 8-bit boundary. This, of course, implies that a frame may be N-bits in length. (The vast majority of applications tend to use frames which are multiples of 8 bits long, however.)

The fact that the flag has a unique binary pattern would seem to limit the contents of the frame since a flag pattern might inadvertently occur within the frame. This would cause the receiver to think the closing flag was received, invalidating the frame. SDLC handles this situation through a technique called zero bit insertion. This techniques specifies that within a frame a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1s. Thus, no pattern of 01111110 is ever transmitted by chance. On the receiving end, after the opening flag is detected, the receiver removes any 0 following 5 consecutive 1s. The inserted and deleted 0s are not counted for error determination.

Before discussing the address field, an explanation of the roles of an SDLC station is in order. SDLC specifies two types of stations: primary and secondary. The primary is the control station for the data link and thus has responsibility of the overall network. There is only one predetermined primary station, all other stations on the link assume the secondary station role. In general, a secondary station speaks only when spoken to. In other words, the primary polls the secondaries for responses. In order to specify a specific secondary, each secondary is assigned a unique 8-bit address. It is this address that is used in the frame's address field.

When the primary transmits a frame to a specific secondary, the address field contains the secondary's address. When responding, the secondary uses its own address in the address field. The primary is never identified. This ensures that the primary knows which of many secondaries is responding since the primary may have many messages outstanding at various secondary stations. In addition to the specific secondary address, an address common to all secondaries may be used for various purposes. (An all 1s address field is usually used for this "All Parties" address.) Even though the primary may use this common address, the secondaries are expected to respond with their unique address. The address field is always the first 8 bits following the opening flag.

The 8 bits following the address field form the control field. The control field embodies the link-level control of SDLC. A detailed explanation of the commands and responses contained in this field is beyond the scope of this application note. Suffice it to say that it is in the control field that the implied acknowledgement is carried out through the use of frame sequence numbers. None of the currently available SDLC single chip controllers utilize the control field. They simply pass it to the processor for analysis. Readers wishing a more detailed explanation of the control field, or of SDLC in general, should consult the IBM documents referenced on the front page overleaf.

In some types of frames, an information field follows the control field. Frames used strictly for link management may or may not contain one. When an information field is used, it is unrestricted in both content and length. This code transparency is made possible because of the zero bit insertion mentioned earlier and the bit-oriented nature of SDLC. Even main memory core dumps may be transmitted because of this capability. This feature is unique to bit-oriented protocols. Like the control field, the information field is not interpreted by the SDLC device; it is merely transferred to and from memory to be operated on and interpreted by the processor.

The final field is the frame check sequence (FCS). The FCS is the 16 bits immediately preceding the closing flag. This 16-bit field is used for error detection through a Cyclic Redundancy Checkword (CRC). The 16-bit transmitted CRC is the complement of the remainder obtained when the A, C, and I fields are "divided" by a generating polynomial. The receiver accumulates the A, C, and I fields and also the FCS into its internal CRC register. At the closing flag, this register contains one particular number for an error-free reception. If this number is not obtained, the frame was received in error and should be discarded. Discarding the frame causes the station to not update its frame sequence numbering. This results in a retransmission after the station sends an acknowledgement from previous frames. [Unlike all other fields, the FCS is transmitted MSB (Most Significant Bit) first. The A, C, and I fields are transmitted LSB (Least Significant Bit) first.] The details of how the FCS is generated and checked is beyond the scope of this application note and since all single component SDLC controllers handle this function automatically, it is usually sufficient to know only that an error has or has not occurred. The IBM documents contain more detailed information for those readers desiring it.

The closing flag terminates the frame. When the closing flag is received, the receiver knows that the preceding 16 bits constitute the FCS and that any bits between the control field and the FCS constitute the information field.

SDLC does not support an interframe time-fill character such as the SYN character in Bi-Sync. If an unusual condition occurs while transmitting, such as data is not available in time from memory or CTS (Clear-to-Send) is lost from the modem, the transmitter aborts the frame by sending an Abort character to notify the receiver to invalidate the frame. The Abort character consists of eight contiguous 1s sent without zero bit insertion. Intraframe time-fill consists of either flags, Abort characters, or any combination of the two.

While the Abort character protects the receiver from transmitted errors, errors introduced by the transmission medium are discovered at the receiver through the FCS check and a check for invalid frames. Invalid frames are those which are not bounded by flags or are too short, that is, less than 32 bits between flags. All invalid frames are ignored by the receiver.

Although SDLC is a synchronous protocol, it provides an optional feature that allows its use on basically asynchronous data links—NRZI (Non-Return-to-Zero-Inverted) coding. NRZI coding specifies that the signal condition does not change for transmitting a binary 1, while a binary 0 causes a change of state. Figure 2 illustrates NRZI coding compared to the normal NRZ. NRZI coding guarantees that an active line will have a transition at least every 5-bit times; long strings of zeroes cause a transition every bit time, while long strings of 1s are broken up by zero bit insertion. Since asynchronous operation requires that the receiver sampling clock be derived from the received data, NRZI encoding plus zero bit insertion make the design of clock recovery circuitry easier.

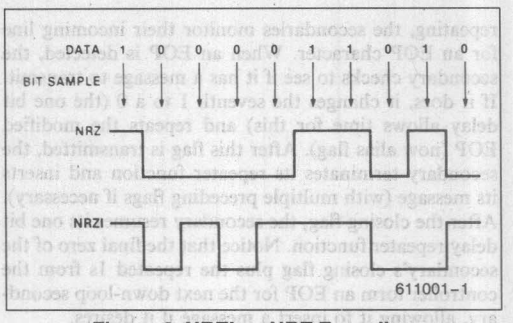


Figure 2. NRZI vs NRZ Encoding

All of the previous discussion has applied to SDLC on either point-to-point or multi-point data networks. SDLC (but not HDLC) also includes specification for a loop configuration. Figure 3 compares these three configurations. IBM uses this loop configuration in its 3650 Retail Store System. It consists of a single loop controller station with one or more down-loop secondary stations. Communications on a loop rely on the secondary stations repeating a received message down loop with a delay of one bit time. The reason for the one bit delay will be evident shortly.

Loop operation defines a new special character: the EOP (End-of-Poll) character which consists of a 0 followed by 7 contiguous, non-zero bit inserted, ones. After the loop controller transmits a message, it idles the line (sends all 1s). The final zero of the closing flag plus the first 7 1s of the idle form an EOP character. While

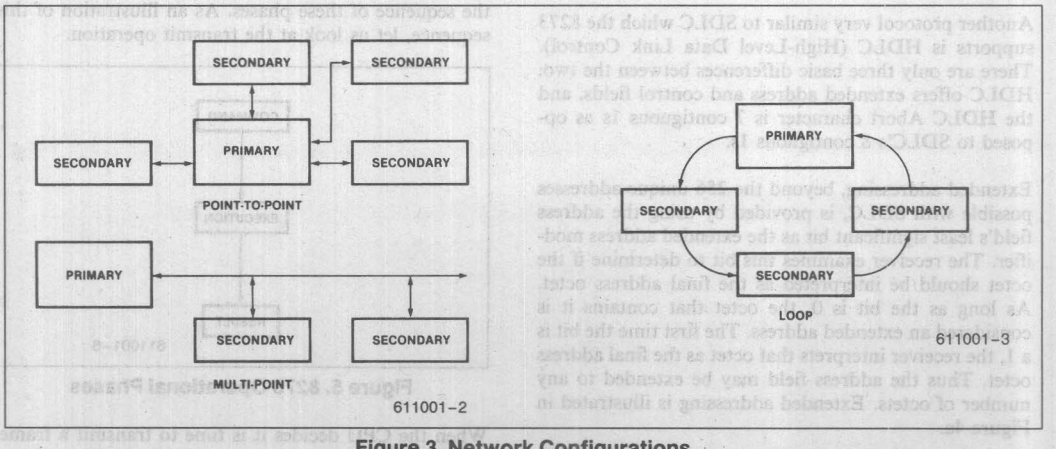


Figure 3. Network Configurations

repeating, the secondaries monitor their incoming line for an EOP character. When an EOP is detected, the secondary checks to see if it has a message to transmit. If it does, it changes the seventh 1 to a 0 (the one bit delay allows time for this) and repeats the modified EOP (now alias flag). After this flag is transmitted, the secondary terminates its repeater function and inserts its message (with multiple preceding flags if necessary). After the closing flag, the secondary resumes its one bit delay repeater function. Notice that the final zero of the secondary's closing flag plus the repeated 1s from the controller form an EOP for the next down-loop secondary, allowing it to insert a message if it desires.

One might wonder if the secondary missed any messages from the controller while it was inserting its own message. It does not. Loop operation is basically half-duplex. The controller waits until it receives an EOP before it transmits its next message. The controller's reception of the EOP signifies that the original message has propagated around the loop followed by any messages inserted by the secondaries. Notice that secondaries cannot communicate with one another directly, all secondary-to-secondary communication takes place by way of the controller.

Loop protocol does not utilize the normal Abort character. Instead, an abort is accomplished by simply transmitting a flag character. Down loop, the receiver sees the abort as a frame which is either too short (if the abort occurred early in the frame) or one with an FCS error. Either results in a discarded frame. For more details on loop operation, please refer to the IBM documents referenced earlier.

Another protocol very similar to SDLC which the 8273 supports is HDLC (High-Level Data Link Control). There are only three basic differences between the two: HDLC offers extended address and control fields, and the HDLC Abort character is 7 contiguous 1s as opposed to SDLC's 8 contiguous 1s.

Extended addressing, beyond the 256 unique addresses possible with SDLC, is provided by using the address field's least significant bit as the extended address modifier. The receiver examines this bit to determine if the octet should be interpreted as the final address octet. As long as the bit is 0, the octet that contains it is considered an extended address. The first time the bit is a 1, the receiver interprets that octet as the final address octet. Thus the address field may be extended to any number of octets. Extended addressing is illustrated in Figure 4a.

A similar technique is used to extend the control field although the extension is limited to only one extra control octet. Figure 4b illustrates control field extension.

Those readers not yet asleep may have noticed the similarity between the SDLC loop EOP character (a 0 followed by 7 1s) and the HDLC Abort (7 1s). This possible incompatibility is neatly handled by the HDLC protocol not specifying a loop configuration.

This completes our brief discussion of the SDLC/HDLC protocols. Now let us turn to the 8273 in particular and discuss its hardware aspects through an explanation of the block diagram and generalized system schematics.

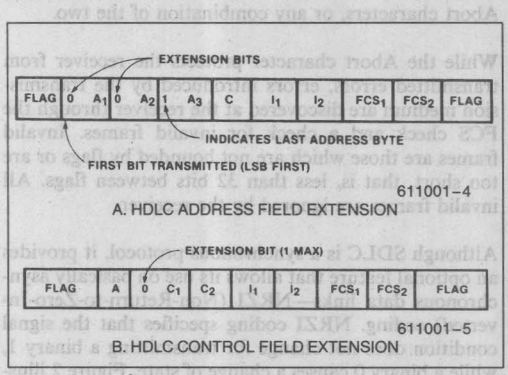


Figure 4

BASIC 8273 OPERATION

It will be helpful for the following discussions to have some idea of the basic operation of the 8273. Each operation, whether it is a frame transmission, reception or port read, etc., is comprised of three phases: the Command, Execution, and Result phases. Figure 5 shows the sequence of these phases. As an illustration of this sequence, let us look at the transmit operation.

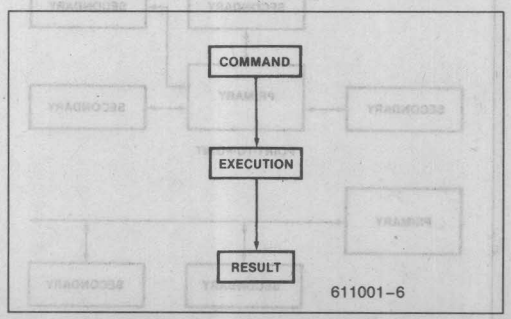


Figure 5. 8273 Operational Phases

When the CPU decides it is time to transmit a frame, the Command phase is entered by the CPU issuing a Transmit Frame command to the 8273. It is not sufficient to just instruct the 8273 to transmit. The frame level command structure sometimes requires more information such as frame length and address and control field content. Once this additional information is sup-

plied, the Command phase is complete and the Execution phase is entered. It is during the Execution phase that the actual operation, in this case a frame transmission, takes place. The 8273 transmits the opening flag, A and C fields, the specified number of I field bytes, inserts the FCS, and closes with the closing flag. Once the closing flag is transmitted, the 8273 leaves the Execution phase and begins the Result phase. During the Result phase the 8273 notifies the CPU of the outcome of the command by supplying interrupt results. In this case, the results would be either that the frame is complete or that some error condition causes the transmission to be aborted. Once the CPU reads all of the results (there is only one for the Transmit Frame command), the Result phase and consequently the operation, is complete. Now that we have a general feeling for the operation of the 8273, let us discuss the 8273 in detail.

HARDWARE ASPECTS OF THE 8273

The 8273 block diagram is shown in Figure 6. It consists of two major interfaces: the CPU module interface and the modem interface. Let's discuss each interface separately.

CPU Interface

The CPU interface consists of four major blocks: Control/Read/Write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

The CPU module utilizes the C/R/W logic to issue commands to the 8273. Once the 8273 receives a command and executes it, it returns the results (good/bad completion) of the command by way of the C/R/W logic. The C/R/W logic is supported by seven registers which are addressed via the A_0, A_1, $\overline{RD}$, and $\overline{WR}$ signals, in addition to $\overline{CS}$. The A_0 and A_1 signals are generally derived from the two low order bits of the CPU module address bus while $\overline{RD}$ and $\overline{WR}$ are the normal I/O Read and Write signals found on the system control bus. Figure 7 shows the address of each register using the C/R/W logic. The function of each register is defined as follows:

Address Inputs		Control Inputs	
A_1	A_0	$\overline{CS}\bullet\overline{RD}$	$\overline{CS}\bullet\overline{WR}$
0	0	Status	Command
0	1	Result	Parameter
1	0	TxI/R	Test Mode
1	1	RxI/R	—

Figure 7. 8273 Register Selection

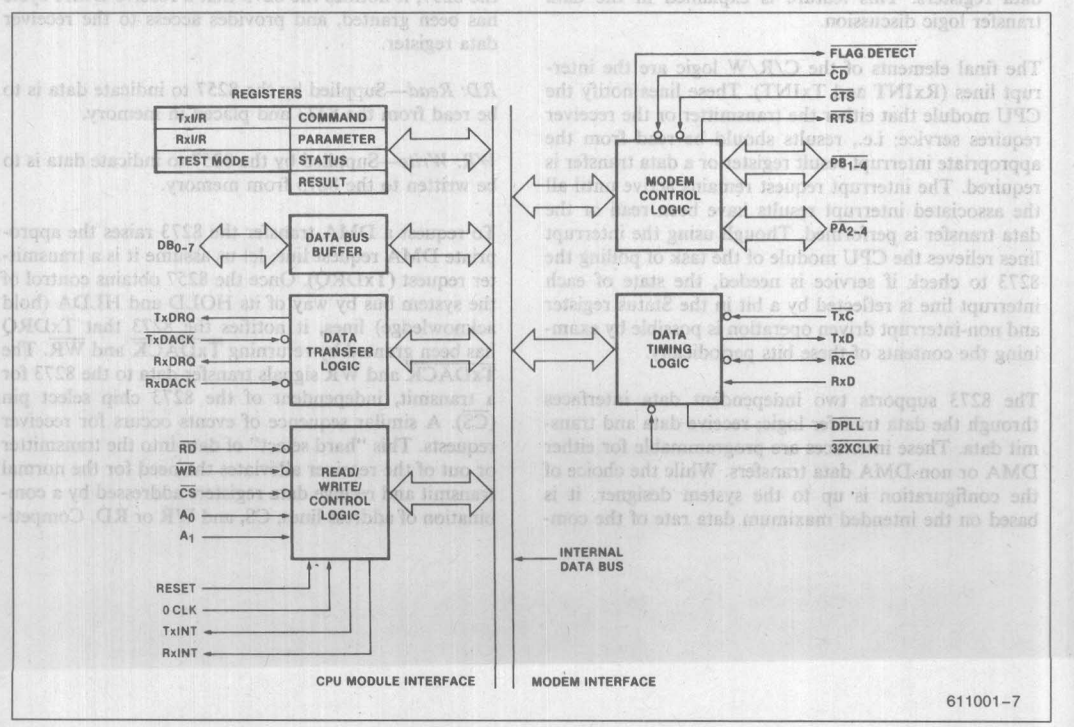


Figure 6. 8273 Block Diagram

Command—8273 operations are initiated by writing the appropriate command byte into this register.

Parameter—Many commands require more information than found in the command itself. This additional information is provided by way of the parameter register.

Immediate Result (Result)—The completion information (results) for commands which execute immediately are provided in this register.

Transmit Interrupt Result (TxI/R)—Results of transmit operations are passed to the CPU in this register.

Receiver Interrupt Result (RxI/R)—Receive operation results are passed to the CPU via this register.

Status—The general status of the 8273 is provided in this register. The Status register supplies the handshaking necessary during various phases of the 8273 operation.

Test Mode—This register provides a software reset function for the 8273.

The commands, parameters, and bit definition of these registers are discussed in the following software section. Notice that there are not specific transmit or receive data registers. This feature is explained in the data transfer logic discussion.

The final elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). These lines notify the CPU module that either the transmitter or the receiver requires service; i.e., results should be read from the appropriate interrupt result register or a data transfer is required. The interrupt request remains active until all the associated interrupt results have been read or the data transfer is performed. Though using the interrupt lines relieves the CPU module of the task of polling the 8273 to check if service is needed, the state of each interrupt line is reflected by a bit in the Status register and non-interrupt driven operation is possible by examining the contents of these bits periodically.

The 8273 supports two independent data interfaces through the data transfer logic; receive data and transmit data. These interfaces are programmable for either DMA or non-DMA data transfers. While the choice of the configuration is up to the system designer, it is based on the intended maximum data rate of the communications channel. Figure 8 illustrates the transfer rate of data bytes that are acquired by the 8273 based on link data rate. Full-duplex data rates above 9600 baud usually require DMA. Slower speeds may or may not require DMA depending on the task load and interrupt response time of the processor.

Figure 9 shows the 8273 in a typical DMA environment. Notice that a separate DMA controller, in this case the Intel 8257, is required. The DMA controller supplies the timing and addresses for the data transfers while the 8273 manages the requesting of transfers and the actual counting of the data block lengths. In this case, elements of the data transfer interface are:

TxDRQ: Transmit DMA Request—Asserted by the 8273, this line requests a DMA transfer from memory to the 8273 for transmit.

TxDACK: Transmit DMA Acknowledge—Returned by the 8257 in response to TxDRQ, this line notifies the 8273 that a request has been granted, and provides access to the transmitter data register.

RxDRQ: Receive DMA Request—Asserted by the 8273, it requests a DMA transfer from the 8273 to memory for a receive operation.

RxDACK: Receive DMA Acknowledge—Returned by the 8257, it notifies the 8273 that a receive DMA cycle has been granted, and provides access to the receiver data register.

RD: Read—Supplied by the 8257 to indicate data is to be read from the 8273 and placed in memory.

WR: Write—Supplied by the 8257 to indicate data is to be written to the 8273 from memory.

To request a DMA transfer the 8273 raises the appropriate DMA request line; let us assume it is a transmitter request (TxDRQ). Once the 8257 obtains control of the system bus by way of its HOLD and HLDA (hold acknowledge) lines, it notifies the 8273 that TxDRQ has been granted by returning $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$. The $\overline{\text{TxDACK}}$ and $\overline{\text{WR}}$ signals transfer data to the 8273 for a transmit, independent of the 8273 chip select pin ($\overline{\text{CS}}$). A similar sequence of events occurs for receiver requests. This "hard select" of data into the transmitter or out of the receiver alleviates the need for the normal transmit and receive data registers addressed by a combination of address lines, CS, and WR or RD. Competi-

tive devices that do not have this "hard select" feature require the use of an external multiplexer to supply the correct inputs for register selection during DMA. (Do not forget that the SDLC controller sees both the addresses and control signals supplied by the DMA controller during DMA cycles.) Let us look at typical frame transmit and frame receive sequences to better see how the 8273 truly manages the DMA data transfer.

Before a frame can be transmitted, the DMA controller is supplied, by the CPU, the starting address for the desired information field. The 8273 is then commanded to transmit a frame. (Just how this is done is covered later during our software discussion.) After the command, but before transmission begins, the 8273 needs a little more information (parameters). Four parameters are required for the transmit frame command: the address field byte, the control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (Request-to-Send) active and waits for CTS (Clear-to-Send) to go active. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field; it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point the requests stop, the FCS and closing flag are transmitted, and the TxINT line is raised, signaling the CPU that the frame transmission is complete. Notice that after the initial command and parameter loading, absolutely no CPU intervention was required (since DMA is used for data transfers) until the entire frame was transmitted. Now let's look at a frame reception.

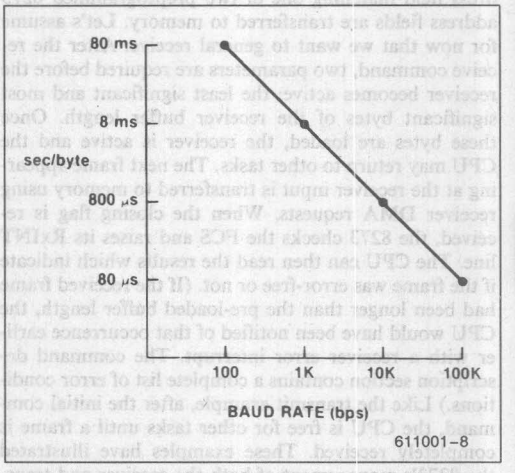


Figure 8. Byte Transfer Rate vs Baud Rate

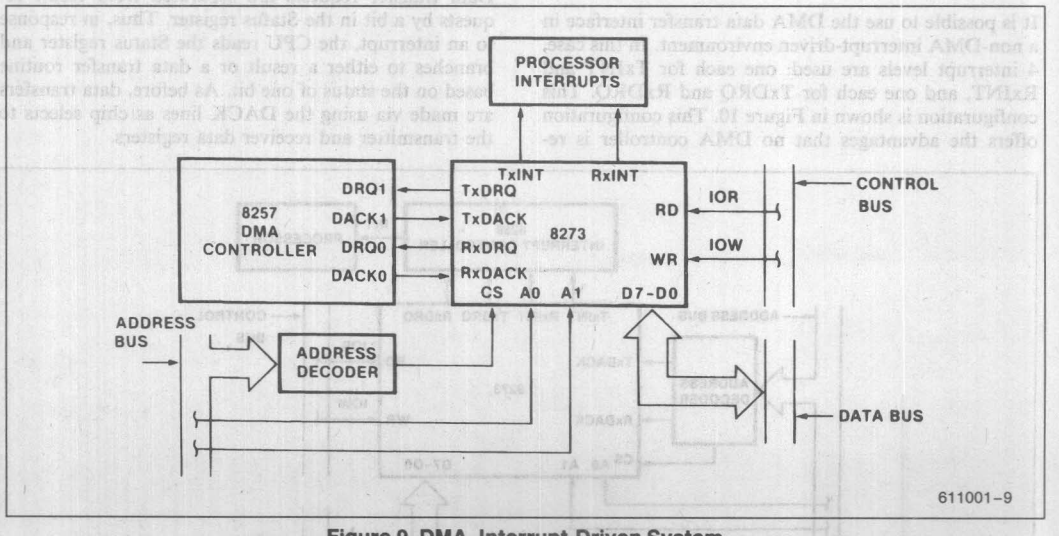


Figure 9. DMA, Interrupt-Driven System

The receiver operation is very similar. Like the initial transmit sequence, the DMA controller is loaded with a starting address for a receiver data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands: General Receive, where all received frames are transferred to memory, and Selective Receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory. Let's assume for now that we want to general receive. After the receive command, two parameters are required before the receiver becomes active: the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the CPU may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The CPU can then read the results which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the CPU would have been notified of that occurrence earlier with a receiver error interrupt. The command description section contains a complete list of error conditions.) Like the transmit example, after the initial command, the CPU is free for other tasks until a frame is completely received. These examples have illustrated the 8273's management of both the receiver and transmitter DMA channels.

It is possible to use the DMA data transfer interface in a non-DMA interrupt-driven environment. In this case, 4 interrupt levels are used: one each for TxINT and RxINT, and one each for TxDRQ and RxDRQ. This configuration is shown in Figure 10. This configuration offers the advantages that no DMA controller is required and data requests are still separated from result (completion) requests. The disadvantages of the configuration are that 4 interrupt levels are required and that the CPU must actually supply the data transfers. This, of course, reduces the maximum data rate compared to the configuration based strictly on DMA. This system could use an Intel 8259 8-level Priority Interrupt Controller to supply a vectored CALL (subroutine) address based on requests on its inputs. The 8273 transmitter and receiver make data requests by raising the respective DRQ line. The CPU is interrupted by the 8259 and vectored to a data transfer routine. This routine either writes (for transmit) or reads (for receive) the 8273 using the respective TxDACK or RxDACK line. The DACK lines serve as "hard" chip selects into and out of the 8273. $\overline{\text{TxDACK}} + \overline{\text{WR}}$ writes data into the 8273 for transmit. $\overline{\text{RxDACK}} + \overline{\text{RD}}$ reads data from the 8273 for receive.) The CPU is notified of operation completion and results by way of TxINT and RxINT lines. Using the 8273, and the 8259, in this way, provides a very effective, yet simple, interrupt-driven interface.

Figure 11 illustrates a system very similar to that described above. This system utilizes the 8273 in a non-DMA data transfer mode as opposed to the two DMA approaches shown in Figures 9 and 10. In the non-DMA case, data transfer requests are made on the TxINT and RxINT lines. The DRQ lines are not used. Data transfer requests are separated from result requests by a bit in the Status register. Thus, in response to an interrupt, the CPU reads the Status register and branches to either a result or a data transfer routine based on the status of one bit. As before, data transfers are made via using the DACK lines as chip selects to the transmitter and receiver data registers.

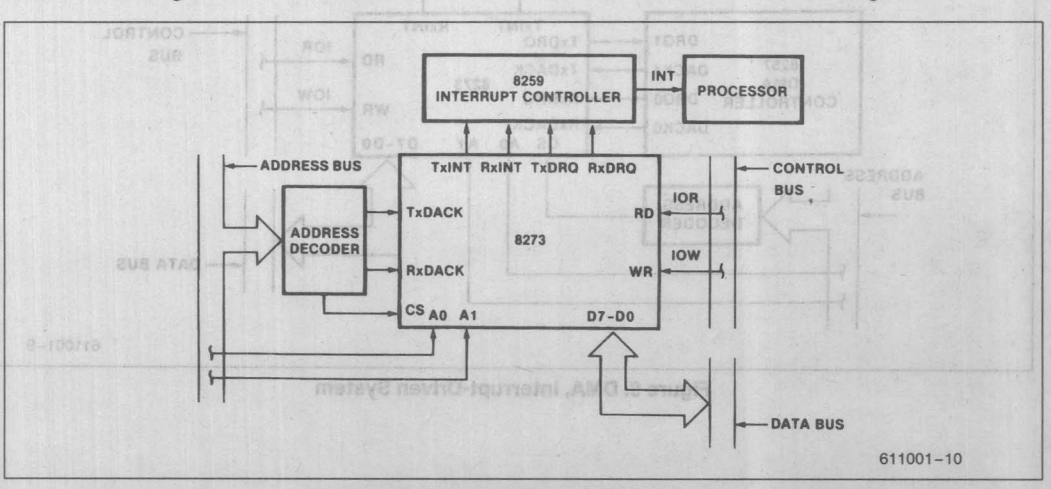


Figure 10. Interrupt-Based DMA System

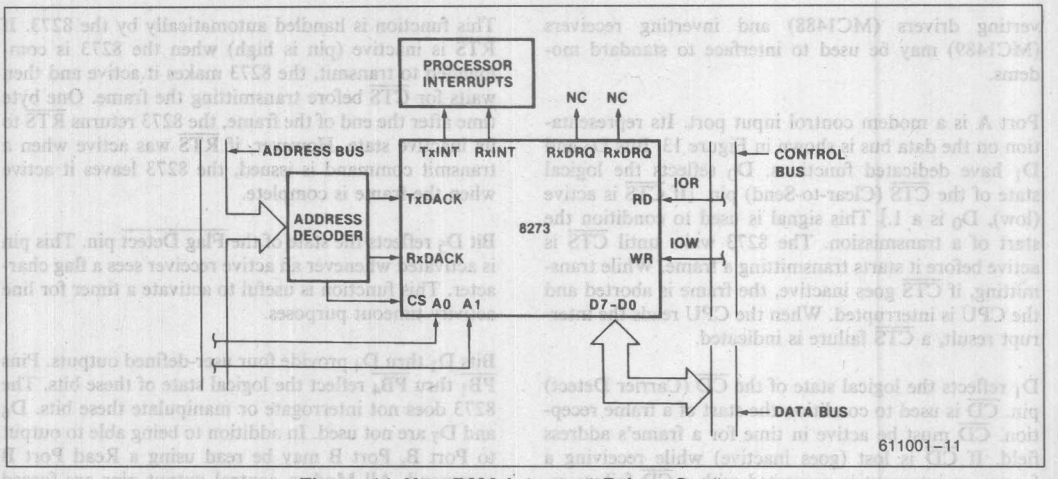


Figure 11. Non-DMA Interrupt-Driven System

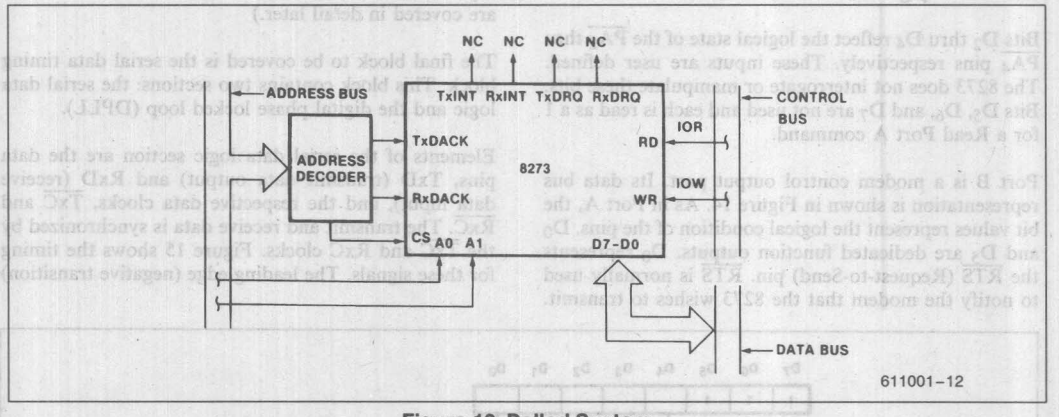


Figure 12. Polled System

Figure 12 illustrates the simplest system of all. This system utilizes polling for all data transfers and results. Since the interrupt pins are reflected in bits in the Status register, the software can read the Status register periodically looking for one of these to be set. If it finds an INT bit set, the appropriate Result Available bit is examined to determine if the "interrupt" is a data transfer or completion result. If a data transfer is called for, the DACK line is used to enter or read the data from the 8273. If the interrupt is a completion result, the appropriate result register is read to determine the good/bad completion of the operation.

The actual selection of either DMA or non-DMA modes is controlled by a command issued during initialization. This command is covered in detail during the software discussion.

The final block of the CPU module interface is the Data Bus Buffer. This block supplies the tri-state, bidirectional data bus interface to allow communication to and from the 8273.

Modem Interface

As the name implies, the modem interface is the modem side of the 8273. It consists of two major blocks: the modem control block and the serial data timing block.

The modem control block provides both dedicated and user-defined modem control functions. All signals supported by this interface are active low so that EIA in-

verting drivers (MC1488) and inverting receivers (MC1489) may be used to interface to standard modems.

Port A is a modem control input port. Its representation on the data bus is shown in Figure 13. Bits D_0 and D_1 have dedicated functions. D_0 reflects the logical state of the $\overline{CTS}$ (Clear-to-Send) pin. [If $\overline{CTS}$ is active (low), D_0 is a 1.] This signal is used to condition the start of a transmission. The 8273 waits until $\overline{CTS}$ is active before it starts transmitting a frame. While transmitting, if $\overline{CTS}$ goes inactive, the frame is aborted and the CPU is interrupted. When the CPU reads the interrupt result, a $\overline{CTS}$ failure is indicated.

D_1 reflects the logical state of the $\overline{CD}$ (Carrier Detect) pin. $\overline{CD}$ is used to condition the start of a frame reception. $\overline{CD}$ must be active in time for a frame's address field. If $\overline{CD}$ is lost (goes inactive) while receiving a frame, an interrupt is generated with a $\overline{CD}$ failure result. $\overline{CD}$ may go inactive between frames.

Bits D_2 thru D_4 reflect the logical state of the $\overline{PA_2}$ thru $\overline{PA_4}$ pins respectively. These inputs are user defined. The 8273 does not interrogate or manipulate these bits. Bits D_5, D_6, and D_7 are not used and each is read as a 1 for a Read Port A command.

Port B is a modem control output port. Its data bus representation is shown in Figure 14. As in Port A, the bit values represent the logical condition of the pins. D_0 and D_5 are dedicated function outputs. D_0 represents the $\overline{RTS}$ (Request-to-Send) pin. $\overline{RTS}$ is normally used to notify the modem that the 8273 wishes to transmit. This function is handled automatically by the 8273. If $\overline{RTS}$ is inactive (pin is high) when the 8273 is commanded to transmit, the 8273 makes it active and then waits for $\overline{CTS}$ before transmitting the frame. One byte time after the end of the frame, the 8273 returns $\overline{RTS}$ to its inactive state. However, if $\overline{RTS}$ was active when a transmit command is issued, the 8273 leaves it active when the frame is complete.

Bit D_5 reflects the state of the $\overline{Flag\ Detect}$ pin. This pin is activated whenever an active receiver sees a flag character. This function is useful to activate a timer for line activity timeout purposes.

Bits D_1 thru D_4 provide four user-defined outputs. Pins $\overline{PB_1}$ thru $\overline{PB_4}$ reflect the logical state of these bits. The 8273 does not interrogate or manipulate these bits. D_6 and D_7 are not used. In addition to being able to output to Port B, Port B may be read using a Read Port B command. All Modem control output pins are forced high on reset. (All commands mentioned in this section are covered in detail later.)

The final block to be covered is the serial data timing block. This block contains two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins, TxD (transmit data output) and RxD (receive data input), and the respective data clocks, $\overline{TxC}$ and $\overline{RxC}$. The transmit and receive data is synchronized by the $\overline{TxC}$ and $\overline{RxC}$ clocks. Figure 15 shows the timing for these signals. The leading edge (negative transition)

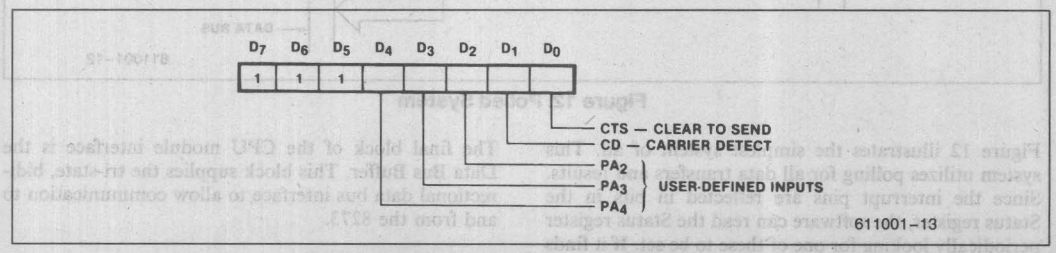


Figure 13. Port A (Input) Bit Definition

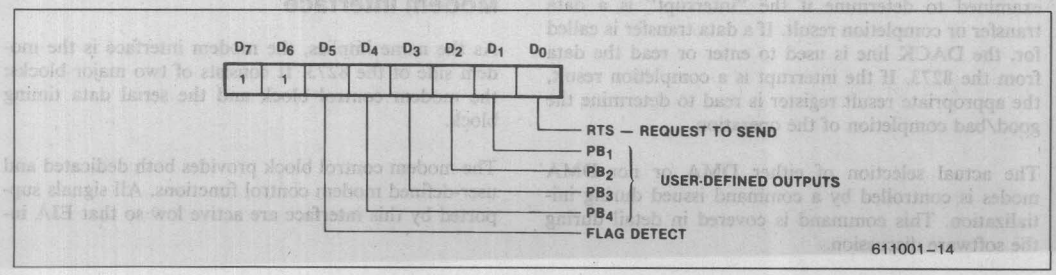


Figure 14. Port B (Output) Bit Definition

of $\overline{TxC}$ generates new transmit data and the trailing edge (positive transition) of $\overline{RxC}$ is used to capture the receive data.

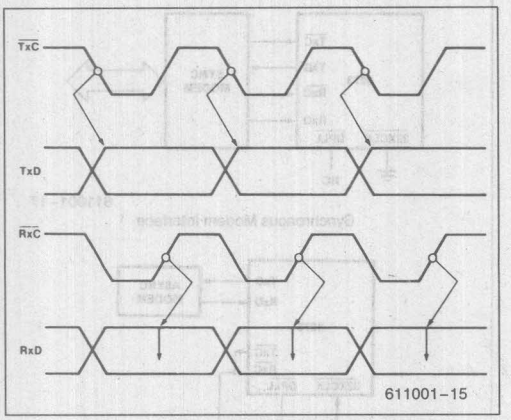


Figure 15. Transmit/Receive Timing

It is possible to reconfigure this section under program control to perform diagnostic functions; both data and clock loopback are available. In data loopback mode, the TxD pin is internally routed to the RxD pin. This allows simple board checkout since the CPU can send an SDLC message to itself. (Note that transmitted data will still appear on the TxD pin.)

When data loopback is utilized, the receiver may be presented incorrect sample timing ($\overline{RxC}$) by the external circuitry. Clock loopback overcomes this problem by allowing the internal routing of $\overline{TxC}$ and $\overline{RxC}$. Thus the same clock used to transmit the data is used to receive it. Examination of Figure 15 shows that this method ensures bit synchronism. The final element of the serial data logic is the Digital Phase Locked Loop.

The DPLL provides a means of clock recovery from the received data stream. This feature allows the 8273 to interface without external synchronizing logic to low cost asynchronous modems (modems which do not supply clocks). It also makes the problem of clock timing in loop configurations trivial.

To use the DPLL, a clock at 32 times the required baud rate must be supplied to the $\overline{32\times\ CLK}$ pin. This clock provides the interval that the DPLL samples the received data. The DPLL uses the 32× clock and the received data to generate a pulse at the $\overline{DPLL}$ output pin. This $\overline{DPLL}$ pulse is positioned at the nominal center of the received data bit cell. Thus the $\overline{DPLL}$ output may be wired to $\overline{RxC}$ and/or $\overline{TxC}$ to supply the data timing. The exact position of the pulse is varied depending on the line noise and bit distortion of the received data. The adjustment of the $\overline{DPLL}$ position is determined according to the rules outlined in Figure 16.

Adjustments to the sample phase of $\overline{DPLL}$ with respect to the received data is made in discrete increments. Referring to Figure 16, following the occurrence of $\overline{DPLL}$

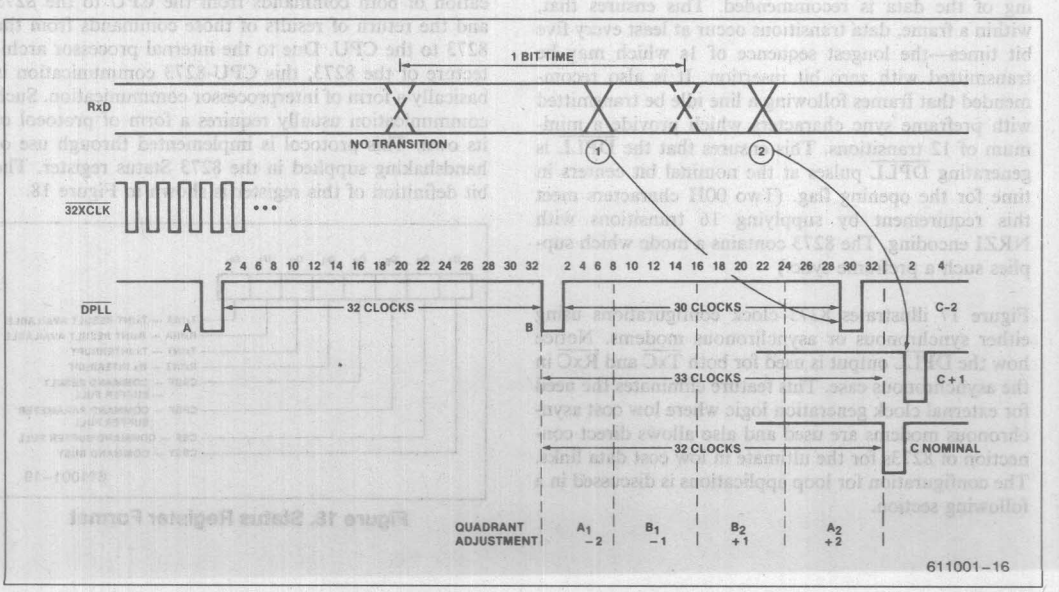


Figure 16. DPLL Phase Adjustments

pulse A, the DPLL counts $\overline{32\times\ CLK}$ pulses and examines the received data for a data edge. Should no edge be detected in 32 pulses, the $\overline{DPLL}$ positions the next $\overline{DPLL}$ pulse (B) at 32 clock pulses from pulse A. Since no new phase information is contained in the data stream, the sample phase is assumed to be at nominal $1\times$ baud rate. Now assume a data edge occurs after $\overline{DPLL}$ pulse B. The distance from B to the next pulse C is influenced according to which quadrant (A_1, B_1, B_2, or A_2) the data edge falls in. (Each quadrant represents 8 $\overline{32\times\ CLK}$ times.) For example, if the edge is detected in quadrant A_1, it is apparent that pulse B was too close to the data edge and the time to the next pulse must be shortened. The adjustment for quadrant A_1 is specified as -2. Thus, the next $\overline{DPLL}$ pulse, pulse C, is positioned 32 $-$ 2 or 30 $\overline{32\times\ CLK}$ pulses following $\overline{DPLL}$ pulse B. This adjustment moves pulse C closer to the nominal bit center of the next received data cell. A data edge occurring in quadrant B_2 would have caused the adjustment to be small, namely 32 + 1 or 33 $\overline{32\times\ CLK}$ pulses. Using this technique, the $\overline{DPLL}$ pulse converges to the nominal bit center within 12 data transitions, worse case—4-bit times adjusting through quadrant A_1 or A_2 and 8-bit times adjusting through B_1 or B_2.

When the receive data stream goes idle after 15 ones, DPLL pulses are generated at 32 pulse intervals of the $32\times$ CLK. This feature allows the DPLL pulses to be used as both transmitter and receiver clocks.

In order to guarantee sufficient transitions of the received data to enable the $\overline{DPLL}$ to lock, NRZI encoding of the data is recommended. This ensures that, within a frame, data transitions occur at least every five bit times—the longest sequence of 1s which may be transmitted with zero bit insertion. It is also recommended that frames following a line idle be transmitted with preframe sync characters which provide a minimum of 12 transitions. This ensures that the $\overline{DPLL}$ is generating $\overline{DPLL}$ pulses at the nominal bit centers in time for the opening flag. (Two 00H characters meet this requirement by supplying 16 transitions with NRZI encoding. The 8273 contains a mode which supplies such a preframe sync.)

Figure 17 illustrates 8273 clock configurations using either synchronous or asynchronous modems. Notice how the $\overline{DPLL}$ output is used for both $\overline{TxC}$ and $\overline{RxC}$ in the asynchronous case. This feature eliminates the need for external clock generation logic where low cost asynchronous modems are used and also allows direct connection of 8273s for the ultimate in low cost data links. The configuration for loop applications is discussed in a following section.

This completes our discussion of the hardware aspects of the 8273. Its software aspects are now discussed.

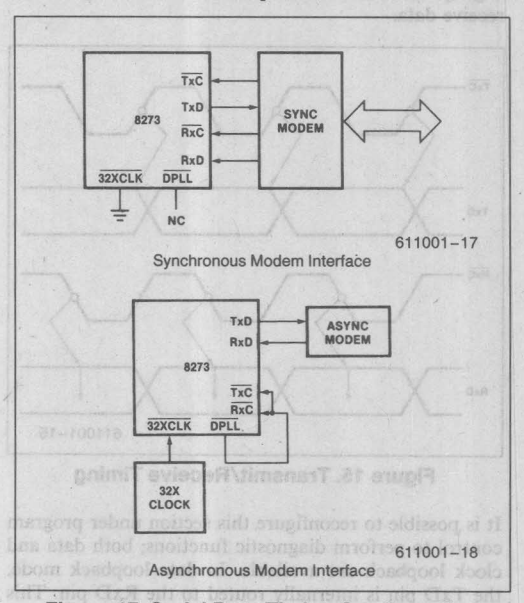


Figure 17. Serial Data Timing Configuration

SOFTWARE ASPECTS OF THE 8273

The software aspects of the 8273 involve the communication of both commands from the CPU to the 8273 and the return of results of those commands from the 8273 to the CPU. Due to the internal processor architecture of the 8273, this CPU-8273 communication is basically a form of interprocessor communication. Such communication usually requires a form of protocol of its own. This protocol is implemented through use of handshaking supplied in the 8273 Status register. The bit definition of this register is shown in Figure 18.

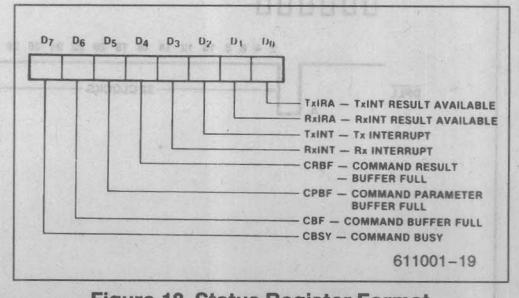


Figure 18. Status Register Format

CBSY: Command Busy—CBSY indicates when the 8273 is in the command phase. CBSY is set when the CPU writes a command into the Command register, starting the Command phase. It is reset when the last parameter is deposited in the Parameter register and accepted by the 8273, completing the Command phase.

CBF: Command Buffer Full—When set, this bit indicates that a byte is present in the Command register. This bit is normally not used.

CPBF: Command Parameter Buffer Full—This bit indicates that the Parameter register contains a parameter. It is set when the CPU deposits a parameter in the Parameter register. It is reset when the 8273 accepts the parameter.

CRBF: Command Result Buffer Full—This bit is set when the 8273 places a result from an immediate type command in the Result register. It is reset when the CPU reads the result from the Result register.

RxINT: Receiver Interrupt—The state of the RxINT pin is reflected by this bit. RxINT is set by the 8273 whenever the receiver needs servicing. RxINT is reset when the CPU reads the results or performs the data transfer.

TxINT: Transmitter Interrupt—This bit is identical to RxINT except action is initiated based on transmitter interrupt sources.

RxIRA: Receiver Interrupt Result Available—RxIRA is set when the 8273 places an interrupt result byte into the RxI/R register. RxIRA is reset when the CPU reads the RxI/R register.

TxIRA: Transmitter Interrupt Result Available—TxIRA is the corresponding Result Available bit for the transmitter. It is set when the 8273 places an interrupt result byte in the TxI/R register and reset when the CPU reads the register.

The significance of each of these bits will be evident shortly. Since the software requirements of each 8273 phase are essentially independent, each phase is covered separately.

Command Phase Software

Recalling the Command phase description in an earlier section, the CPU starts the Command phase by writing a command byte into the 8273 Command register. If further information about the command is required by the 8273, the CPU writes this information into the Parameter register. Figure 19 is a flowchart of the Command phase. Notice that the CBSY and CPBF bits of the Status register are used to handshake the command and parameter bytes. Also note that the chart shows

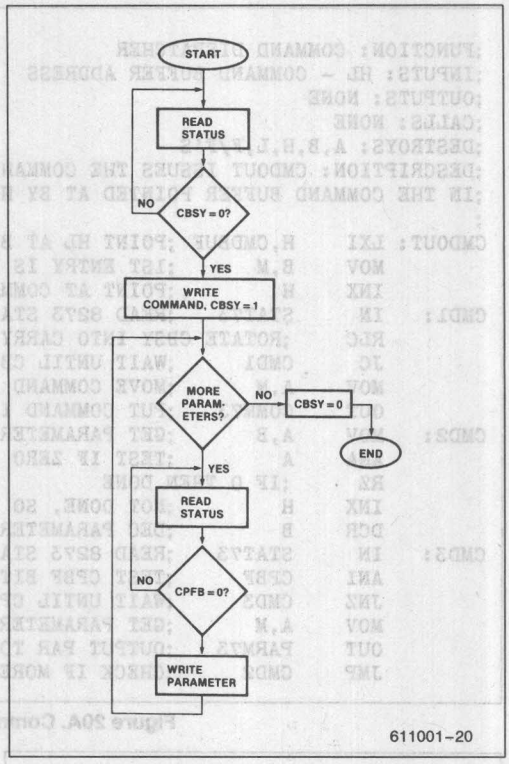


Figure 19. Command Phase Flowchart

that a command may not be issued if the Status register indicates the 8273 is busy (CBSY = 1). If a command is issued while CBSY = 1, the original command is overwritten and lost. (Remember that CBSY signifies the command phase is in progress and not the actual execution of the command.) The flowchart also includes a Parameter buffer full check. The CPU must wait until CPBF = 0 before writing a parameter to the Parameter register. If a parameter is issued while CPBF = 1, the previous parameter is overwritten and lost. An example of command output assembly language software is provided in Figure 20a. This software assumes that a command buffer exists in memory. The buffer is pointed at by the HL register. Figure 20b shows the command buffer structure.

The 8273 is a full duplex device, i.e., both the transmitter and receiver may be executing commands or passing interrupt results at any given time. (Separate Rx and Tx interrupt pins and result registers are provided for this reason.) However, there is only one Command register. Thus, the Command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command phase. A detailed description of the commands and their parameters is presented in a following section.

```
;FUNCTION: COMMAND DISPATCHER
;INPUTS: HL - COMMAND BUFFER ADDRESS
;OUTPUTS: NONE
;CALLS: NONE
;DESTROYS: A,B,H,L,F/F'S
;DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
;IN THE COMMAND BUFFER POINTED AT BY HL
;
CMDOUT: LXI     H,CMDBUF  ;POINT HL AT BUFFER
        MOV     B,M       ;1ST ENTRY IS PAR. COUNT
        INX     H         ;POINT AT COMMAND BYTE
CMD1:   IN      STAT73    ;READ 8273 STATUS
        RLC     ;ROTATE CBSY INTO CARRY
        JC      CMD1      ;WAIT UNTIL CBSY=0
        MOV     A,M       ;MOVE COMMAND BYTE TO A
        OUT     COMM73    ;PUT COMMAND IN COMMAND REG
CMD2:   MOV     A,B       ;GET PARAMETER COUNT
        ANA     A         ;TEST IF ZERO
        RZ      ;IF 0 THEN DONE
        INX     H         ;NOT DONE, SO POINT AT NEXT PAR
        DCR     B         ;DEC PARAMETER COUNT
CMD3:   IN      STAT73    ;READ 8273 STATUS
        ANI     CPBF      ;TEST CPBF BIT
        JNZ     CMD3      ;WAIT UNTIL CPBF IS 0
        MOV     A,M       ;GET PARAMETER FROM BUFFER
        OUT     PARM73    ;OUTPUT PAR TO PARAMETER REG
        JMP     CMD2      ;CHECK IF MORE PARAMETERS
```

Figure 20A. Command Phase Software

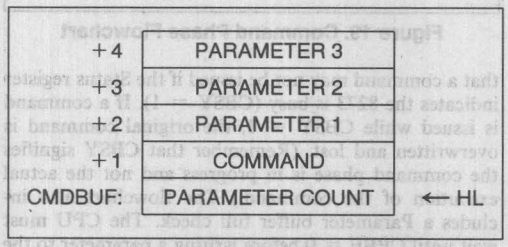


Figure 20B. Command Buffer Format

Execution Phase Software

During the Execution phase, the operation specified by the Command phase is performed. If the system utilizes DMA for data transfers, there is no CPU involvement during this phase, so no software is required. If non-DMA data transfers are used, either interrupts or polling is used to signal a data transfer request.

For interrupt-driven transfers the 8273 raises the appropriate INT pin. When responding to the interrupt,

the CPU must determine whether it is a data transfer request or an interrupt signaling that an operation is complete and results are available. The CPU determines the cause by reading the Status register and interrogating the associated IRA (Interrupt Result Available) bit (TxIRA for TxINT and RxIRA for RxINT). If the IRA = 0, the interrupt is a data transfer request. If the IRA = 1, an operation is complete and the associated Interrupt Result register must be read to determine the completion status (good/bad/etc.). A software interrupt handler implementing the above sequence is presented as part of the Result phase software.

When polling is used to determine when data transfers are required, the polling routine reads the Status register looking for one of the INT bits to be set. When a set INT bit is found, the corresponding IRA bit is examined. Like in the interrupt-driven case, if the IRA = 0, a data transfer is required. If IRA = 1, an operation is complete and the Interrupt Result register needs to be read. Again, example polling software is presented in the next section.

Result Phase Software

During the Result phase the 8273 notifies the CPU of the outcome of a command. The Result phase is initiated by either a successful completion of an operation or an error detected during execution. Some commands such as reading or writing the I/O ports provide immediate results, that is, there is essentially no delay from the issuing of the command and when the result is available. Other commands such as frame transmit, take time to complete so their result is not available immediately. Separate result registers are provided to distinguish these two types of commands and to avoid interrupt handling for simple results.

Immediate results are provided in the Result register. Validity of information in this register is indicated to the CPU by way of the CRBF bit in the Status register. When the CPU completes the Command phase of an immediate command, it polls the Status register waiting until CRBF = 1. When this occurs, the CPU may read the Result register to obtain the immediate result. The Result register provides only the results from immediate commands.

Example software for handling immediate results is shown in Figure 21. The routine returns with the result in the accumulator. The CPU then uses the result as is appropriate.

All non-immediate commands deal with either the transmitter or receiver. Results from these commands are provided in the TxI/R (Transmit Interrupt Result) and RxI/R (Receive Interrupt Result) registers respectively. Results in these registers are conveyed to the CPU by the TxIRA and RxIRA bits of the status register. Results of non-immediate commands consist of one byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The interrupt codes and the meaning of the additional results are covered following the detailed command description.

Non-immediate results are passed to the CPU in response to either interrupts or polling of the Status register. Figure 22 illustrates an interrupt-driven result handler. (Please note that all of the software presented in this application note is not optimized for either speed or code efficiency. They are provided as a guide and to illustrate concepts.) This handler provides for interrupt-driven data transfers as was promised in the last section. Users employing DMA-based transfers do not

```
;FUNCTION: IMDRLT
;INPUTS: NONE
;OUTPUTS: RESULT REGISTER IN A
;CALLS: NONE
;DESTROYS: A, F/F'S
;DESCRIPTION: IMDRLT IS CALLED AFTER A CMDOUT FOR AN
;IMMEDIATE COMMAND TO READ THE RESULT REGISTER
;
IMDRLT: IN    STAT 73   ;READ 8273 STATUS
        ANI   CRBF      ;TEST IF RESULT REG READY
        JZ    IMDRLT    ;WAIT IF CRBF=0
        IN    RESL73    ;READ RESULT REGISTER
        RET   ;RETURN
```

Figure 21. Immediate Result Handler

```
;FUNCTION: RXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: RCRBUF, RCVPNT
;CALLS: NONE
;OUTPUTS: RCRBUF, RCVPNT
;DESTROYS: NOTHING
;DESCRIPTION: RXI IS ENTERED AT A RECEIVER INTERRUPT.
;THE INTERRUPT IS TESTED FOR DATA TRANSFER (IRA=0)
;OR RESULT (IRA=1).  FOR DATA TRANSFER, THE DATA IS
;PLACED IN A BUFFER AT RCVPNT. RESULTS ARE PLACED IN
;A BUFFER AT RCRBUF.
;A FLAG(RXFLAG) IS SET IF THE INTERRUPT WAS A RESULT.
;(DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*) AND
;MAYBE ELIMINATED BY USERS USING DMA.
;
RXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        PUSH    B       ;SAVE B
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     RXIRA   ; (*) TEST IRA BIT
        JZ      RXI2    ; (*) IF 0, DATA TRANSFER NEEDED
RXI1:   LHLD    RCRBUF  ;GET RESULT BUFFER POINTER
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXINT   ;TEST INT BIT
        JZ      RXI4    ;IF 0, THEN DONE
        IN      STAT73  ;READ 8273 STATUS AGAIN
        ANI     RXIRA   ;TEST IRA AGAIN
        JZ      RXI1    ;LOOP UNTIL RESULT IS READY
        IN      RXIR73  ;READY, READ RXI/R
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    RCRBUF  ;RESTORE BUFFER POINTER
        JMP     RXI1    ;GO BACK TO SEE IF MORE
RXI2:   SHLD    RCVPNT  ; (*) GET DATA BUFFER POINTER
        IN      RCVDAT  ; (*) READ DATA VIA RXDACK
        MOV     M,A     ; (*) STORE DATA IN BUFFER
        INX     H       ; (*) BUMP DATA POINTER
        JMP     RXI3    ; (*) DONE
RXI4:   MVI     A,01H   ;SET RX FLAG TO SHOW COMPLETION
        STA     RXFLAG  ;COMPLETION
RXI3:   POP     B       ;RESTORE BC
        POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE

;FUNCTION: TXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: TXRBUF, TXPNT, TXFLAG
;OUTPUTS: TXRBUF, TXPNT, TXFLAG
;CALLS: NONE
;DESTROYS: NOTHING
;DESCRIPTION: TXI IS ENTERED AT A TRANSMITTER INTERRUPT.
;THE INTERRUPT IS TESTED BY WAY OF THE IRA BIT TO SEE
;IF A DATA TRANSFER OR RESULT COMPLETION HAS OCCURED.
;FOR DATA TRANSFERS (IRA=0), THE DATA IS OBTAINED FROM
;A BUFFER LOCATION POINTED AT BY TXPNT.  FOR COMPLETION,
;(IRA=1), THE RESULTS ARE READ AND PLACED AT A RESULT
;BUFFER POINTED AT BY TXRBUF, AND THE TXFLAG IS SET
;TO INDICATE TO THE MAIN PROGRAM THAT A OPERATION IS
;COMPLETE. TX OPERATIONS HAVE ONLY ONE RESULT.
;DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*). THESE
;MAYBE REMOVED BY USERS USING DMA.
;
TXI:    PUSH    H       ;SAVE HL
        PUSH    PSW     ;SAVE PSW
        IN      STAT73  ; (*) READ 8273 STATUS
        ANI     TXIRA   ; (*) TEST TXIRA BIT
        JZ      TXI2    ; (*) IF 0, DATA TRANSFER
        IN      TXIR73  ;1, THEN READ TXIR
        LHLD    TXRBUF  ;GET RESULT BUFFER POINTER
        MOV     M,A     ;STORE RESULT IN BUFFER
        INX     H       ;BUMP RESULT POINTER
        SHLD    TXRBUF  ;RESTORE RESULT POINTER
        MVI     A,01H   ;SET TXFLAG TO SHOW COMPLETION
        STA     TXFLAG  ;SET FLAG
TXI1:   POP     PSW     ;RESTORE PSW
        POP     H       ;RESTORE HL
        EI      ;ENABLE INTERRUPTS
        RET     ;DONE
TXI2:   LHLD    TXPNT   ; (*) GET DATA POINTER
        MOV     A,M     ; (*) GET DATA FROM BUFFER
        OUT     TXDATA  ; (*) OUTPUT TO 8273 VIA TXDACK
        INX     H       ; (*) BUMP DATA POINTER
        SHLD    TXPNT   ; (*) RESTORE POINTER
        JMP     TXI1    ; (*) RETURN AFTER RESTORE
```

611001-61

Figure 22. Interrupt-Driven Result Handlers with Non-DMA Data Transfers

need the lines where the IRA bit is tested for zero. (These lines are denoted by an asterisk in the comments column.) Note that the INT bit is used to determine when all results have been read. All results must be read. Otherwise, the INT bit (and pin) will remain high and further interrupts may be missed. These routines place the results in a result buffer pointed at by RCRBUF and TxRBUF.

A typical result handler for systems utilizing polling is shown in Figure 23. Data transfers are also handled by this routine. This routine utilizes the routines of Figure 22 to handle the results.

At this point, the reader should have a good conceptual feel about how the 8273 operates. It is now time for the particulars of each command to be discussed.

```
;FUNCTION: POLOP
;INPUTS: NONE
;OUTPUTS: C=0 (NO STATUS), =1 (RX COMPLETION),
;   =2 (TX COMPLETION), =3 (BOTH)
;CALLS: TXI, RXI
;DESTROYS: B,C
;DESCRIPTION: POLOP IS CALLED TO POLL THE 8273 FOR
;DATA TRANSFERS AND COMPLETION RESULTS.  THE
;ROUTINES TXI AND RXI ARE USED FOR THE ACTUAL
;TRANSFERS AND BUFFER WORK.  POLOP RETURNS
;THE STATUS OF THEIR ACTION.
;
POLOP:  PUSH    PSW     ;SAVE PSW
        MVI     C,00H   ;CLEAR C
POLOP1: IN      STAT73  ;READ 8273 STATUS
        ANI     INT     ;ARE TXINT OR RXINT SET?
        JZ      PEXIT   ;NO, EXIT
        IN      STAT73  ;READ 8273 STATUS
        ANI     RXINT   ;TEST RX INT
        JNZ     RXIC    ;YES, GO SERVICE RX
        CALL    TXI     ;MUST BE TX, GO SERVICE IT
        LDA     TXFLAG  ;GET TX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        INR     C
        JMP     POLOP1  ;TRY AGAIN
;
RXIC:   CALL    RXI     ;GO SERVICE RX
        LDA     RXFLAG  ;GET RX FLAG
        CPI     01H     ;WAS IT A COMPLETION? (01)
        JNZ     PEXIT   ;NO, SO JUST EXIT
        INR     C       ;YES, UPDATE C
        JMP     POLOP1  ;TRY AGAIN
;
PEXIT:  POP     PSW     ;RESTORE PSW
        RET     ;RETURN WITH COMP. STATUS IN C
```

611001-62

Figure 23. Polling Result Handler

8273 COMMAND DESCRIPTION

In this section, each command is discussed in detail. In order to shorten the notation, please refer to the command key in Table 1. The 8273 utilizes five different command types: Initialization/Configuration, Receive, Transmit, Reset, and Modem Control.

Table 1. Command Summary Key

B_0, B_1	—LSB and MSB of Receive Buffer Length
R_0, R_1	—LSB and MSB of Received Frame Length
L_0, L_1	—LSB and MSB of Transmit Frame Length
A_1, A_2	—Match Addresses for Selective Receive
RIC	—Receiver Interrupt Result Code
TIC	—Transmitter Interrupt Result Code
A	—Address Field of Received Frame
C	—Control Field of Received Frame

Initialization/Configuration Commands

The Initialization/Configuration commands manipulate registers internal to the 8273 that define the various operating modes. These commands either set or reset specified bits in the registers depending on the type of command. One parameter is required. Set commands perform a logical OR operation of the parameter (mask) and the internal register. This mask contains 1s where register bits are to be set. A "0" in the mask causes no change in the corresponding register bit. Reset commands perform a logical AND operation of the parameter (mask) and the internal register, i.e., the mask is "0" to reset a register bit and a "1" to cause no change. Before presenting the commands, the register bit definitions are discussed.

Operating Mode Register (Figure 24)

D_7–D_6: *Not Used*—These bits must not be manipulated by any command; i.e., D_7–D_6 must be 0 for the Set command and 1 for the Reset command.

D_5: *HDLC Abort*—When this bit is set, the 8273 will interrupt when 7 1s (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 1s) will cause an interrupt.

D_4: *EOP Interrupt*—Reception of an EOP character (0 followed by 7 1s) will cause the 8273 to interrupt the CPU when this bit is set. Loop controller stations use this mode as a signal that a polling frame has completed the loop. No EOP interrupt is generated when this bit is reset.

D_3: *Early Tx Interrupt*—This bit specifies when the transmitter should generate an end of frame interrupt. If this bit is set, an interrupt is generated when the last data character has been passed to the 8273. If the user software issues another transmit command within two byte times, the final flag interrupt does not occur and the new frame is transmitted with only one flag of separation. If this restriction is not met, more than one flag will separate the frames and a frame complete interrupt is generated after the closing flag. If the bit is reset, only the frame complete interrupt occurs. This bit, when set, allows a single flag to separate consecutive frames.

D_2: *Buffered Address and Control*—When set, the address and control fields of received frames are buffered in the 8273 and passed to the CPU as results after a received frame interrupt (they are not transferred to memory with the information field). On transmit, the A and C fields are passed to the 8273 as parameters. This mode simplifies buffer management. When this bit is reset, the A and C fields are passed to and from memory as the first two data transfers.

D_1: *Preframe Sync*—When set, the 8273 prefaces each transmitted frame with two characters before the opening flag. These two characters provide 16 transitions to allow synchronization of the opposing receiver. To guarantee 16 transitions, the two characters are 55H-55H for non-NRZI mode (see Serial I/O Register description) or 00H-00H for NRZI mode. When reset, no preframe characters are transmitted.

D_0: *Flag Stream*—When set, the transmitter will start sending flag characters as soon as it is idle; i.e., immediately if idle when the command is issued or after a transmission if the transmitter is active when this bit is set. When reset, the transmitter starts sending Idle characters on the next character boundary if idle already, or at the end of a transmission if active.

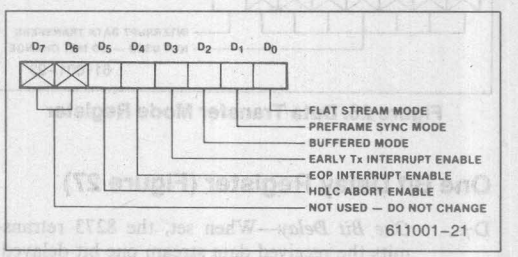


Figure 24. Operating Mode Register

Serial I/O Mode Register (Figure 25)

D_7–D_3: *Not Used*—These bits must be 0 for the Set command and 1 for the Reset command.

D_2: *Data Loopback*—When set, transmitted data (TxD) is internally routed to the receive data circuitry. When reset, TxD and RxD are independent.

D_1: *Clock Loopback*—When set, $\overline{\text{TxC}}$ is internally routed to $\overline{\text{RxC}}$. When reset, the clocks are independent.

D_0: *NRZI (Non-Return to Zero Inverted*—When set, the 8273 assumes the received data is NRZI encoded, and NRZI encodes the transmitted data. When reset, the received and transmitted data are treated as a normal positive logic bit stream.

Data Transfer Mode Register (Figure 26)

D_7–D_1: *Not Used*—These bits must be 0 for the Set command and 1 for the Reset command.

D_0: *Interrupt Data Transfer*—When set, the 8273 will interrupt the CPU when data transfers are required (the corresponding IRA Status register bit will be 0 to signify a data transfer interrupt rather than a Result phase interrupt). When reset, 8273 data transfers are performed through DMA requests on the DRQ pins without interrupting the CPU.

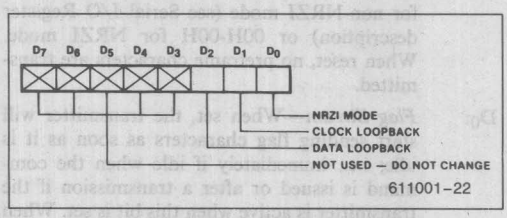


Figure 25. Serial I/O Mode Register

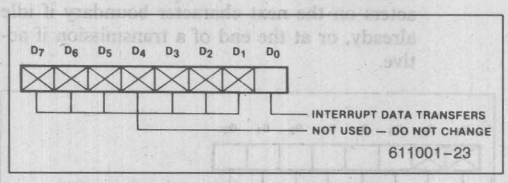


Figure 26. Data Transfer Mode Register

One Bit Delay Register (Figure 27)

D_7: *One Bit Delay*—When set, the 8273 retransmits the received data stream one bit delayed. This mode is entered and exited at a received character boundary. When reset, the transmitted and received data are independent. This mode is utilized for loop operation and is discussed in a later section.

D_6–D_0: *Not Used*—These bit must be 0 for the Set command and 1 for the Reset command.

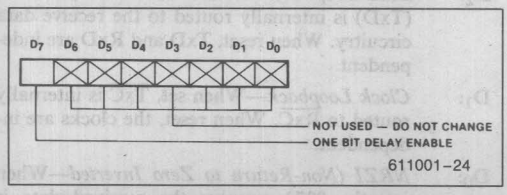


Figure 27. One Bit Delay Mode Register

Figure 28 shows the Set and Reset commands associated with the above registers. The mask which sets or resets the desired bits is treated as a single parameter. These commands do not interrupt nor provide results during the Result phase. After reset, the 8273 defaults to all of these bits reset.

Register	Command	Hex Code	Parameter
One Bit Delay Mode	Set	A4	Set Mask
	Reset	64	Reset Mask
Data Transfer Mode	Set	97	Set Mask
	Reset	57	Reset Mask
Operating Mode	Set	91	Set Mask
	Reset	51	Reset Mask
Serial I/O Mode	Set	A0	Set Mask
	Reset	60	Reset Mask

Figure 28. Initialization/Configuration Command Summary

Receive Commands

The 8273 supports three receive commands plus a receiver disable function.

General Receive

When commanded to General Receive, the 8273 passes all frames either to memory (DMA mode) or to the CPU (non-DMA mode) regardless of the contents of the frame's address field. This command is used for primary and loop controller stations. Two parameters are required: B_0 and B_1. These parameters are the LSB and MSB of the receiver buffer size. Giving the 8273 this extra information alleviates the CPU of the burden of checking for buffer overflow. The 8273 will interrupt the CPU if the received frame attempts to overfill the allotted buffer space.

Selective Receive

In Selective Receive, two additional parameters besides B_0 and B_1 are required: A_1 and A_2. These parameters are two address match bytes. When commanded to Selective Receive, the 8273 passes to memory or the CPU only those frames having an address field matching either A_1 or A_2. This command is usually used for secondary stations with A_1 being the secondary address and A_2 is the "All Parties" address. If only one match byte is needed, A_1 and A_2 should be equal. As in General Receive, the 8273 counts the incoming data bytes and interrupts the CPU if B_0, B_1 is exceeded.

Selective Loop Receive

This command is very similar in operation to Selective Receive except that One Bit Delay mode must be set

and that the loop is captured by placing transmitter in Flag Stream mode automatically after an EOP character is detected following a selectively received frame. The details of using the 8273 in loop configurations is discussed in a later section so please hold questions until then.

The handling of interrupt results is common among the three commands. When a frame is received without error, i.e., the FCS is correct and $\overline{CD}$ (Carrier Detect) was active throughout the frame or no attempt was made to overfill the buffer; the 8273 interrupts the CPU following the closing flag to pass the completion results. These results, in order, are the receiver interrupt result code (RIC), and the byte length of the information field of the received frame (R_0, R_1). If Buffered mode is selected, the address and control fields are passed as two additional results. If Buffered mode is not selected, the address and control fields are passed as the first two data transfers and R_0, R_1 reflect the information field length plus two.

Receive Disable

The receiver may also be disabled using the Receive Disable command. This command terminates any receive operation immediately. No parameters are required and no results are returned.

The details for the Receive command are shown in Figure 29. The interrupt result code key is shown in Figure 30. Some explanation of these result codes is appropriate.

The interrupt result code is the first byte passed to the CPU in the RxI/R register during the Result phase. Bits D_4–D_0 define the cause of the receiver interrupt. Since each result code has specific implications, they are discussed separately below.

2

Command	Hex Code	Parameters	Results* RxI/R
General Receive	C0	B_0, B_1	RIC, R_0, R_1, A, C
Selective Receive	C1	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
Selective Loop Receive	C2	B_0, B_1, A_1, A_2	RIC, R_0, R_1, A, C
Disable Receiver	C5	None	None

*NOTE:
A and C are passed as results only in buffered mode.

Figure 29. Receiver Command Summary

RIC D_7–D_0	Receiver Interrupt Result Code	Rx Status After INT
* 00000	A_1 Match or General Receive	Active
* 00001	A_2 Match	Active
000 00011	CRC Error	Active
000 00100	Abort Detected	Active
000 00101	Idle Detected	Disabled
000 00110	EOP Detected	Disabled
000 00111	Frame < 32 Bits	Active
000 01000	DMA Overrun	Disabled
000 01001	Memory Buffer Overflow	Disabled
000 01010	Carrier Detect Failure	Disabled
000 01011	Receiver Interrupt Overrun	Disabled
*D_7–D_5	**Partial Byte Received**	
111	All 8 Bits of Last Byte	
000	D_0	
100	D_1–D_0	
010	D_2–D_0	
110	D_3–D_0	
001	D_4–D_0	
101	D_5–D_0	
011	D_6–D_0	

Figure 30. Receiver Interrupt Result Codes (RIC)

The first two result codes result from the error-free reception of a frame. If the frame is received correctly after a General Receive command, the first result is returned. If either Selective Receive command was used (normal or loop), a match with A_1 generates the first result code and a match with A_2 generates the second. In either case, the receiver remains active after the interrupt; however, the internal buffer size counters are not reset. That is, if the receive command indicated 100 bytes were allocated to the receive buffer (B_0, B_1) and an 80-byte frame was received correctly, the maximum next frame size that could be received without recommanding the receiver (resetting B_0 and B_1) is 20 bytes. Thus, it is common practice to recommand the receiver after each frame reception. DMA and/or memory pointers are usually updated at this time. (Note that users who do not wish to take advantage of the 8273's buffer management features may simply use B_0, B_1 = 0FFH for each receive command. Then frames of 65K bytes may be received without buffer overflow errors.)

The third result code is a CRC error. This indicates that a frame was received in the correct format (flags, etc.); however, the received FCS did not check with the internally generated FCS. The frame should be discarded. The receiver remains active. (Do not forget that even though an error condition has been detected, all frame information up until that error has either been transferred to memory or passed to the CPU. This information should be invalidated. This applies to all receiver error conditions.) Note that the FCS, either transmitted or received, is never available to the CPU.

The Abort Detect result occurs whenever the receiver sees either an SDLC (8 1s) or an HDLC (7 1s), depending on the Operating Mode register. However, the intervening Abort character between a closing flag and an Idle does not generate an interrupt. If an Abort character (seen by an active receiver within a frame) is not preceded by a flag and is followed by an idle, an interrupt will be generated for the Abort, followed by an Idle interrupt one character time later. The Idle Detect result occurs whenever 15 consecutive 1s are received. After the Abort Detect interrupt, the receiver remains active. After the Idle Detect interrupt, the receiver is disabled and must be recommanded before further frames may be received.

If the EOP Interrupt bit is set in the Operating Mode register, the EOP Detect result is returned whenever an EOP character is received. The receiver is disabled, so the Idle following the EOP does not generate an Idle Detect interrupt.

The minimum number of bits in a valid frame between the flags is 32. Fewer than 32 bits indicates an error. If Buffered mode is selected, such frames are ignored, i.e., no data transfers or interrupts are generated. In non-Buffered mode, a < 32-bit frame generates an interrupt with the < 32-bit frame result since data transfers may already have disturbed the 8257 or interrupt handler. The receiver remains active.

The DMA Overrun results from the $\overline{\text{DMA}}$ controller being too slow in extracting data from the 8273, i.e., the $\overline{\text{RxDACK}}$ signal is not returned before the next received byte is ready for transfer. The receiver is disabled if this error condition occurs.

The Memory Buffer Overflow result occurs when the number of received bytes exceeds the receiver buffer length supplied by the B_0 and B_1 parameters in the receive command. The receiver is disabled.

The Carrier Detect Failure result occurs when the $\overline{\text{CD}}$ pin goes high (inactive) during reception of a frame. The $\overline{\text{CD}}$ pin is used to qualify reception and must be active by the time the address field starts to be received. If $\overline{\text{CD}}$ is lost during the frame, a $\overline{\text{CD}}$ Failure interrupt is generated and the receiver is disabled. No interrupt is generated if $\overline{\text{CD}}$ goes inactive between frames.

If a condition occurs requiring an interrupt be generated before the CPU has finished reading the previous interrupt results, the second interrupt is generated after the current Result phase is complete (the RxINT pin and status bit go low then high). However, the interrupt result for this second interrupt will be a Receive Interrupt Overrun. The actual cause of the second interrupt is lost. One case where this may occur is at the end of a received frame where the line goes idle. The 8273 generates a received frame interrupt after the closing flag and then 15-bit times later, generates an Idle Detect interrupt. If the interrupt service routine is slow in reading the first interrupt's results, the internal RxI/R register still contains result information when the Idle Detect interrupt occurs. Rather than wiping out the previous results, the 8273 adds a Receive Interrupt Overrun result as an extra result. If the system's interrupt structure is such that the second interrupt is not acknowledged (interrupts are still disabled from the first interrupt), the Receive Interrupt Overrun result is read as an extra result, after those from the first interrupt. If the second interrupt is serviced, the Receive Interrupt Overrun is returned as a single result. (Note that the INT pins supply the necessary transitions to support a Programmable Interrupt Controller such as the Intel 8259. Each interrupt generates a positive-going edge on the appropriate INT pin and the high level is held until the interrupt is completely serviced.) In general, it is possible to have interrupts occurring at one character time intervals. Thus the interrupt handling software must have at least that much response and service time.

The occurrence of Receive Interrupt Overruns is an indication of marginal software design; the system's interrupt response and servicing time is not sufficient for the

data rates being attempted. It is advisable to configure the interrupt handling software to simply read the interrupt results, place them into a buffer, and clear the interrupt as quickly as possible. The software can then examine the buffer for new results at its leisure, and take appropriate action. This can easily be accomplished by using a result buffer flag that indicates when new results are available. The interrupt handler sets the flag and the main program resets it once the results are retrieved.

Both SDLC and HDLC allow frames which are of arbitrary length (>32 bits). The 8273 handles this N-bit reception through the high order bits (D_7–D_5) of the result code. These bits code the number of valid received bits in the last received information field byte. This coding is shown in Figure 30. The high order bits of the received partial byte are indeterminate. [The address, control, and information fields are transmitted least significant bit (A_0) first. The FCS is complemented and transmitted most significant bit first.]

Transmit Commands

The 8273 transmitter is supported by three Transmit commands and three corresponding Abort commands.

Transmit Frame

The Transmit Frame command simply transmits a frame. Four parameters are required when Buffered mode is selected and two when it is not. In either case, the first two parameters are the least and the most significant bytes of the desired frame length (L_0, L_1). In Buffered mode, L_0 and L_1 equal the length in bytes of the desired information field, while in the non-Buffered mode, L_0 and L_1 must be specified at the information field length plus two. (L_0 and L_1 specify the number of data transfers to be performed.) In Buffered mode, the address and control fields are presented to the transmitter as the third and fourth parameters respectively. In non-Buffered mode, the A and C fields must be passed as the first two data transfers.

When the Transmit Frame command is issued, the 8273 makes $\overline{RTS}$ (Request-to-Send) active (pin low) if it was not already. It then waits until $\overline{CTS}$ (Clear-to-Send) goes active (pin low) before starting the frame. If the Preframe Sync bit in the Operating Mode register is set, the transmitter prefaces two characters (16 transitions) before the opening flag. If the Flag Stream bit is set in the Operating Mode register, the frame (including Preframe Sync if selected) is started on a flag boundary. Otherwise the frame starts on a character boundary.

At the end of the frame, the transmitter interrupts the CPU (the interrupt results are discussed shortly) and returns to either Idle or Flag Stream, depending on the Flag Stream bit of the Operating Mode register. If $\overline{RTS}$ was active before the transmit command, the 8273 does not change it. If it was inactive, the 8273 will deactivate it within one character time.

Loop Transmit

Loop Transmit is similar to Frame Transmit (the parameter definition is the same). But since it deals with loop configurations, One Bit Delay mode must be selected.

If the transmitter is not in Flag Stream mode when this command is issued, the transmitter waits until after a received EOP character has been converted to a flag (this is done automatically) before transmitting. (The one bit delay is, of course, suspended during transmit.) If the transmitter is already in Flag Stream mode as a result of a selectively received frame during a Selective Loop Receive command, transmission will begin at the next flag boundary for Buffered mode or at the third flag boundary for non-Buffered mode. This discrepancy is to allow time for enough data transfers to occur to fill up the internal transmit buffer. At the end of a Loop Transmit, the One Bit Delay mode is re-entered and the flag stream mode is reset. More detailed loop operation is covered later.

Transmit Transparent

The Transmit Transparent command enables the 8273 to transmit a block of raw data. This data is without SDLC protocol, i.e., no zero bit insertion, flags, or FCS. Thus it is possible to construct and transmit a Bi-Sync message for front-end processor switching or to construct and transmit an SDLC message with incorrect FCS for diagnostic purposes. Only the L_0 and L_1 parameters are used since there are not fields in this mode. (The 8273 does not support a Receive Transparent command.)

Abort Commands

Each of the above transmit commands has an associated Abort command. The Abort Frame Transmit command causes the transmitter to send eight contiguous ones (no zero bit insertion) immediately and then revert to either idle or flag streaming based on the Flag Stream bit. (The 8 1s as an Abort character is compatible with both SDLC and HDLC.)

For Loop Transmit, the Abort Loop Transmit command causes the transmitter to send one flag and then revert to one bit delay. Loop protocol depends upon FCS errors to detect aborted frames.

The Abort Transmit Transparent simply causes the transmitter to revert to either idles or flags as a function of the Flag Stream mode specified.

The Abort commands require no parameters, however, they do generate an interrupt and return a result when complete.

A summary of the Transmit commands is shown in Figure 31. Figure 32 shows the various transmit interrupt result codes. As in the receiver operation, the transmitter generates interrupts based on either good completion of an operation or an error condition to start the Result phase.

The Early Transmit Interrupt result occurs after the last data transfer to the 8273 if the Early Transmit Interrupt bit is set in the Operating Mode register. If the 8273 is commanded to transmit again within two character times, a single flag will separate the frames. (Buffered mode must be used for a single flag to separate the frames. If non-Buffered mode is selected, three flags will separate the frames.) If this time constraint is not met, another interrupt is generated and multiple flags or idles will separate the frames. The second interrupt is the normal Frame Transmit Complete interrupt. The Frame Transmit Complete result occurs at the closing flag to signify a good completion.

The DMA Underrun result is analogous to the DMA Overrun result in the receiver. Since SDLC does not support intraframe time fill, if the DMA controller or CPU does not supply the data in time, the frame must be aborted. The action taken by the transmitter on this error is automatic. It aborts the frame just as if an Abort command had been issued.

Clear-to-Send Error result is generated if $\overline{CTS}$ goes inactive during a frame transmission. The frame is aborted as above.

The Abort Complete result is self-explanatory. Please note however that no Abort Complete interrupt is generated when an automatic abort occurs. The next command type consists of only one command.

Reset Command

The Reset command provides a software reset function for the 8273. It is a special case and does not utilize the normal command interface. The reset facility is provided in the Test Mode register. The 8273 is reset by simply outputting a 01H followed by a 00H to the Test Mode register. Writing the 01 followed by the 00 mimicks the action required by the hardware reset. Since the 8273 requires time to process the reset internally, at least 10 cycles of the ϕCLK clock must occur between the writing of the 01 and the 00. The action taken is the same as if a hardware reset is performed, namely:

1) The modem control outputs are forced high inactive.

Command	Hex Code	Parameters*	Results TxI/R
Transmit Frame	C8	L_0, L_1, A, C	TIC
Abort	CC	None	TIC
Loop Transmit	CA	L_0, L_1, A, C	TIC
Abort	CE	None	TIC
Transmit Transparent	C0	L_0, L_1	TIC
Abort	CD	None	TIC

*NOTE:
A and C are passed as parameters in buffered mode only.

Figure 31. Transmitter Command Summary

RIC D_7–D_0	Transmitter Interrupt Result Code	Tx Status after INT
000 01100	Early Tx Interrupt	Active
000 01101	Frame Tx Complete	Idle or Flags
000 01110	DMA Underrun	Abort
000 01111	Clear to Send Error	Abort
000 10000	Abort Complete	Idle or Flags

Figure 32. Transmitter Interrupt Result Codes

2) The 8273 Status register is cleared.

3) Any commands in progress cease.

4) The 8273 enters an idle state until the next command is issued.

Modem Control Commands

The modem control ports were discussed earlier in the Hardware section. The commands used to manipulate these ports are shown in Figure 33. The Read Port A and Read Port B commands are immediate. The bit definition for the returned byte is shown in Figures 13 and 14. Do not forget that the returned value represents the logical condition of the pin, i.e., pin active (low) = bit set.

The Set and Reset Port B commands are similar to the Initialization commands in that they use a mask parameter which defines the bits to be changed. Set Port B utilizes a logical OR mask and Reset Port B uses a logical AND mask. Setting a bit makes the pin active (low). Resetting the bit deactivates the pin (high).

To help clarify the numerous timing relationships that occur and their consequences, Figures 34 and 35 are provided as an illustration of several typical sequences. It is suggested that the reader go over these diagrams and re-read the appropriate part of the previous sections if necessary.

HDLC CONSIDERATIONS

The 8273 supports HDLC as well as SDLC. Let's discuss how the 8273 handles the three basic HDLC/SDLC differences: extended addressing, extended control, and the 7 1s Abort character.

Recalling Figure 4a, HDLC supports an address field of indefinite length. The actual amount of extension used is determined by the least significant bit of the characters immediately following the opening flag. If the LSB is 0, more address field bytes follow. If the LSB is 1, this byte is the final address field byte. Software must be used to determine this extension.

If non-Buffered mode is used, the A, C, and I fields are in memory. The software must examine the initial characters to find the extent of the address field. If Buffered mode is used, the characters corresponding to the SDLC A and C fields are transferred to the CPU as interrupt results. Buffered mode assumes the two characters following the opening flag are to be transferred as interrupt results regardless of content or meaning. (The 8273 does not know whether it is being used in an SDLC or an HDLC environment.) In SDLC, these characters are necessarily the A and C field bytes, however in HDLC, their meaning may change depending on the amount of extension used. The software must recognize this and examine the transferred results as possible address field extensions.

Frames may still be selectively received as is needed for secondary stations. The Selective Receive command is still used. This command qualifies a frame reception on the first byte following the opening flag matching either of the A_1 or A_2 match byte parameters. While this does not allow qualification over the complete range of HDLC addresses, it does perform a qualification on the first address byte. The remaining address field bytes, if any, are then examined via software to completely qualify the frame.

Once the extent of the address field is found, the following bytes form the control field. The same LSB test used for the address field is applied to these bytes to determine the control field extension, up to two bytes maximum. The remaining frame bytes in memory represent the information field.

The Abort character difference is handled in the Operating Mode register. If the HDLC Abort Enable bit is set, the reception of seven contiguous ones by an active receiver will generate an Abort Detect interrupt rather than eight ones. (Note that both the HDLC Abort Enable bit and the EOP Interrupt bit must not be set simultaneously.)

Now let's move on to the SDLC loop configuration discussion.

Port	Command	Hex Code	Parameter	Reg Result
A Input	Read	22	None	Port Value
B Output	Read	23	None	Port Value
	Set	A3	Set Mask	None
	Reset	63	Reset Mask	None

Figure 33. Modem Control Command Summary

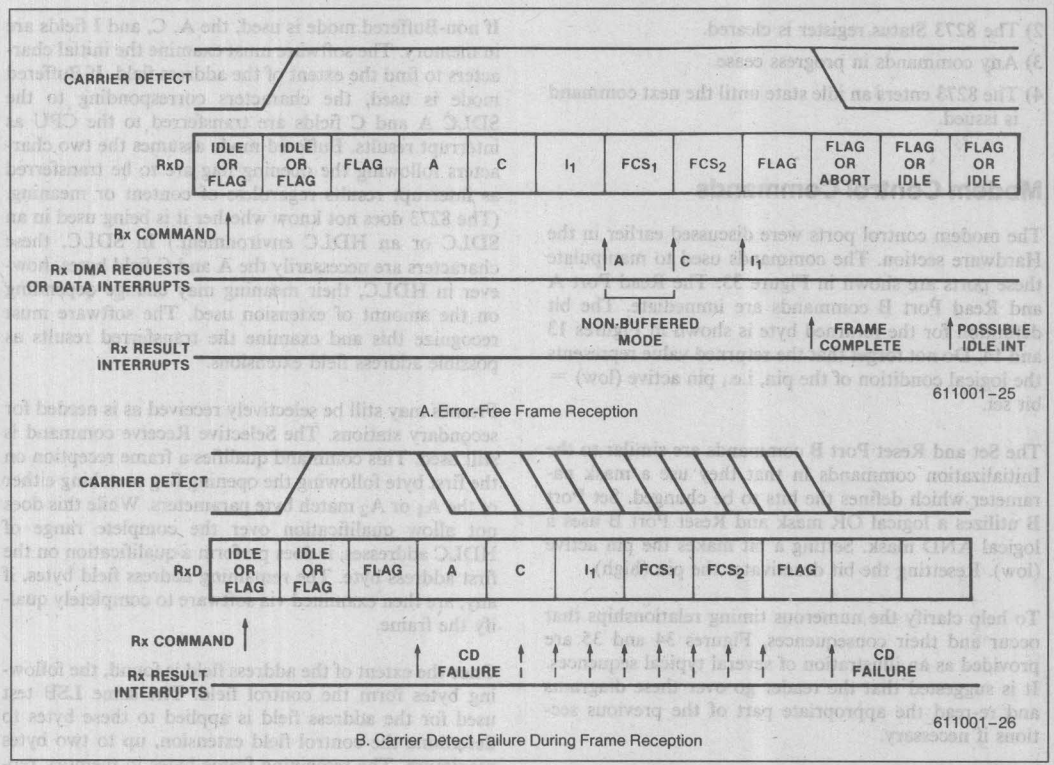


Figure 34. Sample Receiver Timing Diagrams

LOOP CONFIGURATION

Aside from use in the normal data link applications, the 8273 is extremely attractive in loop configuration due to the special frame-level loop commands and the Digital Phase Locked Loop. Toward this end, this section details the hardware and software considerations when using the 8273 in a loop application.

The loop configuration offers a simple, low-cost solution for systems with multiple stations within a small physical location, i.e., retail stores and banks. There are two primary reasons to consider a loop configuration. The interconnect cost is lower for a loop over a multi-point configuration since only one twisted pair or fiber optic cable is used. (The loop configuration does not support the passing of distinct clock signals from station to station.) In addition, loop stations do not need the intelligence of a multi-point station since the loop protocol is simpler. The most difficult aspects of loop station design are clock recovery and implementation of one bit delay (both are handled neatly by the 8273).

Figure 36 illustrates a typical loop configuration with one controller and two down-loop secondaries. Each station must derive its own data timing from the received data stream. Recalling our earlier discussion of the DPLL, notice that $\overline{TxC}$ and $\overline{RxC}$ clocks are provided by the $\overline{DPLL}$ output. The only clock required in the secondaries is a simple, non-synchronized clock at 32 times the desired baud rate. The controller requires both 32× and 1× clocks. (The 1× is usually implemented by dividing the 32× clock with a 5-bit divider. However, there is no synchronism requirement between these clocks so any convenient implementation may be used.)

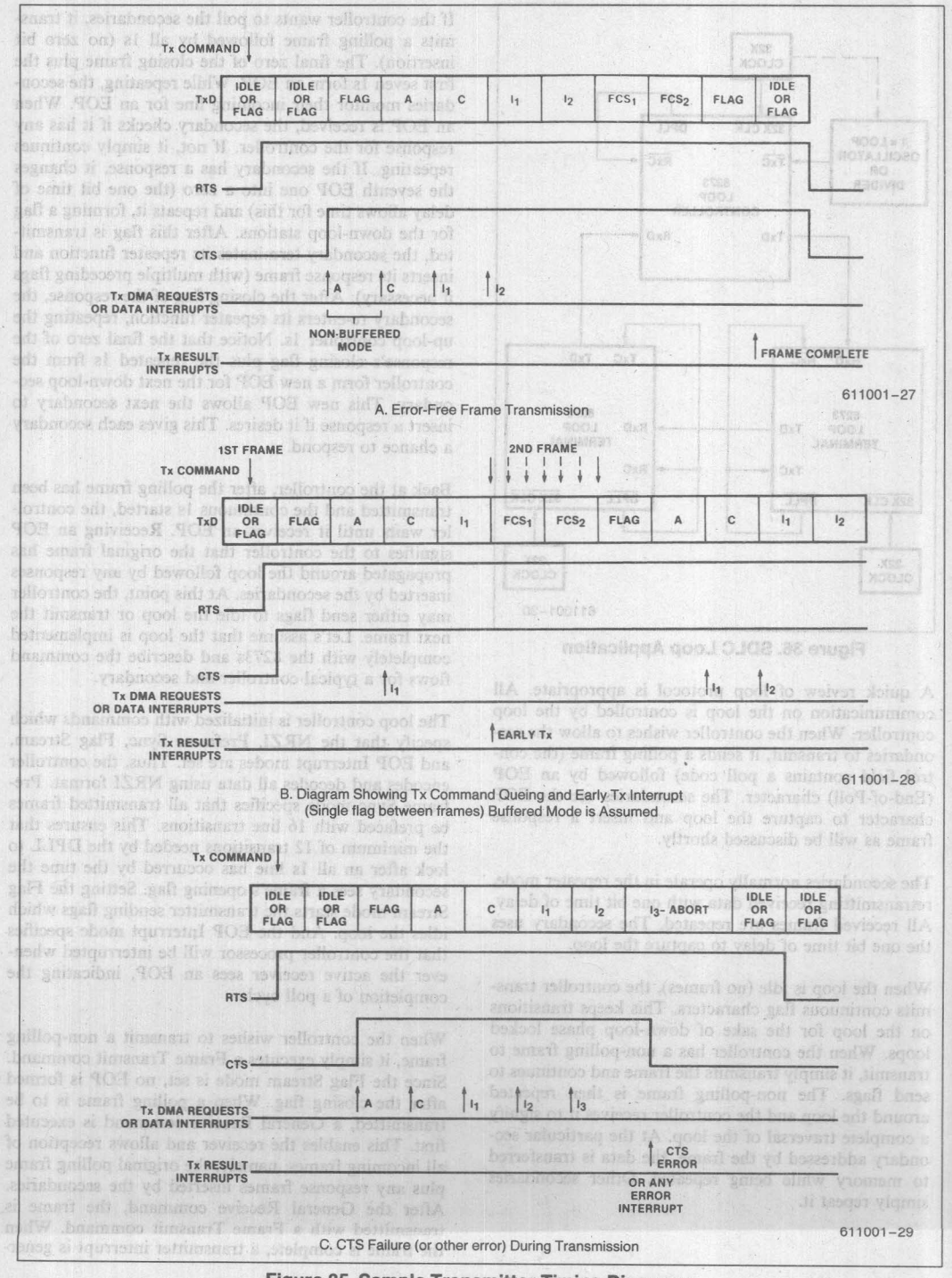


Figure 35. Sample Transmitter Timing Diagrams

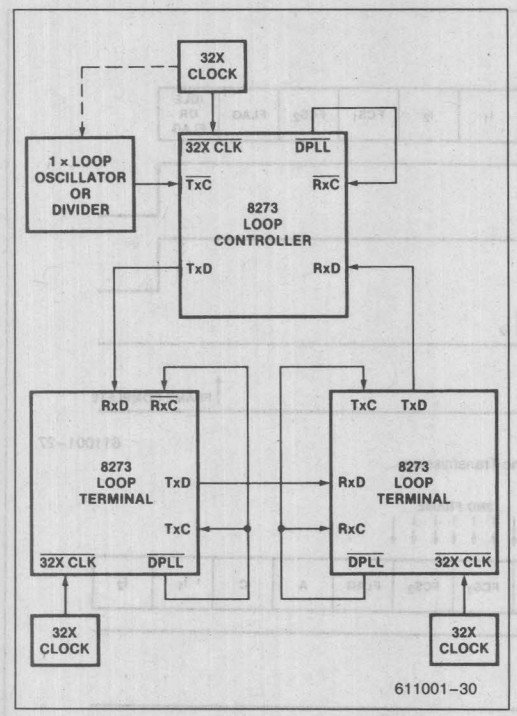


Figure 36. SDLC Loop Application

A quick review of loop protocol is appropriate. All communication on the loop is controlled by the loop controller. When the controller wishes to allow the secondaries to transmit, it sends a polling frame (the control field contains a poll code) followed by an EOP (End-of-Poll) character. The secondaries use the EOP character to capture the loop and insert a response frame as will be discussed shortly.

The secondaries normally operate in the repeater mode, retransmitting received data with one bit time of delay. All received frames are repeated. The secondary uses the one bit time of delay to capture the loop.

When the loop is idle (no frames), the controller transmits continuous flag characters. This keeps transitions on the loop for the sake of down-loop phase locked loops. When the controller has a non-polling frame to transmit, it simply transmits the frame and continues to send flags. The non-polling frame is then repeated around the loop and the controller receives it to signify a complete traversal of the loop. At the particular secondary addressed by the frame, the data is transferred to memory while being repeated. Other secondaries simply repeat it.

If the controller wants to poll the secondaries, it transmits a polling frame followed by all 1s (no zero bit insertion). The final zero of the closing frame plus the first seven 1s form an EOP. While repeating, the secondaries monitor their incoming line for an EOP. When an EOP is received, the secondary checks if it has any response for the controller. If not, it simply continues repeating. If the secondary has a response, it changes the seventh EOP one into a zero (the one bit time of delay allows time for this) and repeats it, forming a flag for the down-loop stations. After this flag is transmitted, the secondary terminates its repeater function and inserts its response frame (with multiple preceding flags if necessary). After the closing flag of the response, the secondary re-enters its repeater function, repeating the up-loop controller 1s. Notice that the final zero of the response's closing flag plus the repeated 1s from the controller form a new EOP for the next down-loop secondary. This new EOP allows the next secondary to insert a response if it desires. This gives each secondary a chance to respond.

Back at the controller, after the polling frame has been transmitted and the continuous 1s started, the controller waits until it receives an EOP. Receiving an EOP signifies to the controller that the original frame has propagated around the loop followed by any responses inserted by the secondaries. At this point, the controller may either send flags to idle the loop or transmit the next frame. Let's assume that the loop is implemented completely with the 8273s and describe the command flows for a typical controller and secondary.

The loop controller is initialized with commands which specify that the NRZI, Preframe Sync, Flag Stream, and EOP Interrupt modes are set. Thus, the controller encodes and decodes all data using NRZI format. Preframe Sync mode specifies that all transmitted frames be prefaced with 16 line transitions. This ensures that the minimum of 12 transitions needed by the DPLL to lock after an all 1s line has occurred by the time the secondary sees a frame's opening flag. Setting the Flag Stream mode starts the transmitter sending flags which idles the loop. And the EOP Interrupt mode specifies that the controller processor will be interrupted whenever the active receiver sees an EOP, indicating the completion of a poll cycle.

When the controller wishes to transmit a non-polling frame, it simply executes a Frame Transmit command. Since the Flag Stream mode is set, no EOP is formed after the closing flag. When a polling frame is to be transmitted, a General Receive command is executed first. This enables the receiver and allows reception of all incoming frames; namely, the original polling frame plus any response frames inserted by the secondaries. After the General Receive command, the frame is transmitted with a Frame Transmit command. When the frame is complete, a transmitter interrupt is gener-

ated. The loop controller processor uses this interrupt to reset Flag Stream mode. This causes the transmitter to start sending all 1s. An EOP is formed by the last flag and the first 7 1s. This completes the loop controller transmit sequence.

At any time following the start of the polling frame transmission the loop controller receiver will start receiving frames. (The exact time difference depends, of course, on the number of down-loop secondaries due to each inserting one bit time of delay.) The first received frame is simply the original polling frame. However, any additional frames are those inserted by the secondaries. The loop controller processor knows all frames have been received when it sees an EOP Interrupt. This interrupt is generated by the 8273 since the EOP Interrupt mode was set during initialization. At this point, the transmitter may be commanded either to enter Flag Stream mode, idling the loop, or to transmit the next frame. A flowchart of this sequence is shown in Figure 37.

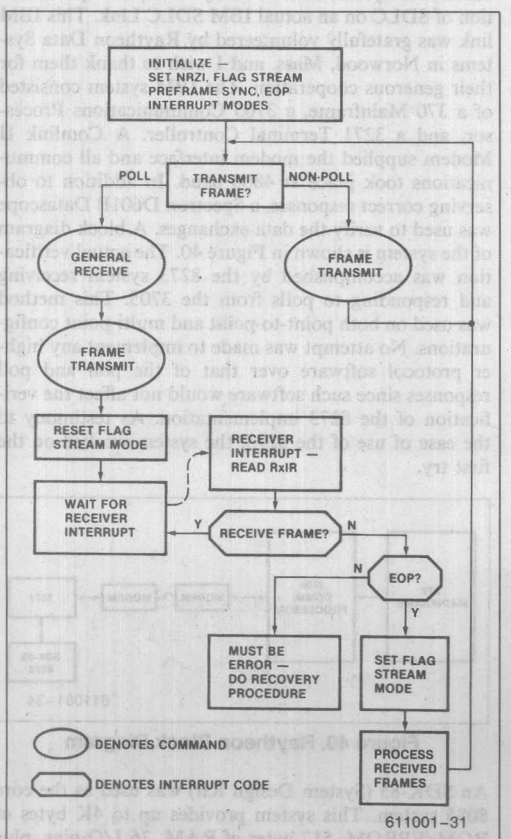


Figure 37. Loop Controller Flowchart

The secondaries are initialized with the NRZI and One Bit Delay modes set. This puts the 8273 into the repeater mode with the transmitter repeating the received data with one bit time of delay. Since a loop station cannot transmit until it sees an EOP character, any transmit command is queued until an EOP is received. Thus whenever the secondary wishes to transmit a response, a Loop Transmit command is issued. The 8273 then waits until it receives an EOP. At this point, the receiver changes the EOP into a flag, repeats it, resets One Bit Delay mode stopping the repeater function, and sets the transmitter into Flag Stream mode. This captures the loop. The transmitter now inserts its message. At the closing flag, Flag Stream mode is reset, and One Bit Delay mode is set, returning the 8273 to repeater function and forming an EOP for the next down-loop station. These actions happen automatically after a Loop Transmit command is issued.

When the secondary wants its receiver enabled, a Selective Loop Receive command is issued. The receiver then looks for a frame having a match in the Address field. Once such a frame is received, repeated, and transferred to memory, the secondary's processor is interrupted with the appropriate Match interrupt result and the 8273 continues with the repeater function until an EOP is received, at which point the loop is captured as above. The processor should use the interrupt to determine if it has a message for the controller. If it does, it simply issues a Loop Transmit command and things progress as above. If the processor has no message, the software must reset the Flag Stream mode bit in the Operating Mode register. This will inhibit the 8273 from capturing the loop at the EOP. (The match frame and the EOP may be separated in time by several frames depending on how many up-loop stations inserted messages of their own.) If the timing is such that the receiver has already captured the loop when the Flag Stream mode bit is reset, the mode is exited on a flag boundary and the frame just appears to have extra closing flags before the EOP. Notice that the 8273 handles the queuing of the transmit commands and the setting and resetting of the mode bits automatically. Figure 38 illustrates the major points of the secondary command sequence.

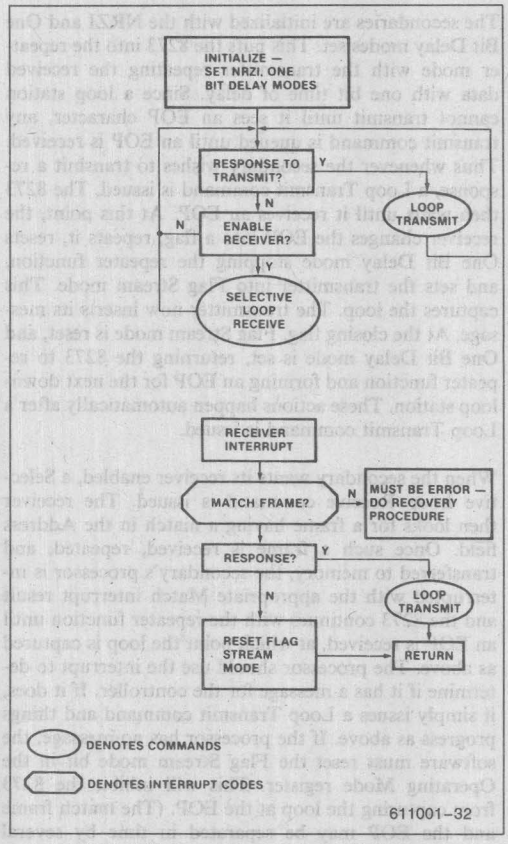


Figure 38. Loop Secondary Flowchart

When an off-line secondary wishes to come on-line, it must do so in a manner which does not disturb data on the loop. Figure 39 shows a typical hardware interface. The line labeled Port could be one of the 8273 Port B outputs and is assumed to be high (1) initially. Thus up-loop data is simply passed down-loop with no delay; however, the receiver may still monitor data on the loop. To come on-line, the secondary is initialized with only the EOP Interrupt mode set. The up-loop data is then monitored until an EOP occurs. At this point, the secondary's CPU is interrupted with an EOP interrupt. This signals the CPU to set One Bit Delay mode in the 8273 and then to set Port low (active). These actions switch the secondary's one bit delay into the loop. Since after the EOP only 1s are traversing the loop, no loop disturbance occurs. The secondary now waits for the next EOP, captures the loop, and inserts a "new on-line" message. This signals the controller that a new secondary exists and must be acknowledged. After the secondary receives its acknowledgement, the normal command flow is used.

It is hopefully evident from the above discussion that the 8273 offers a very simple and easy to implement solution for designing loop stations whether they are controllers or down-loop secondaries.

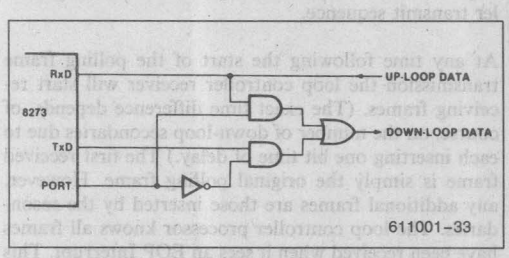


Figure 39. Loop Interface

APPLICATION EXAMPLE

This section describes the hardware and software of the 8273/8085 system used to verify the 8273 implementation of SDLC on an actual IBM SDLC Link. This IBM link was gratefully volunteered by Raytheon Data Systems in Norwood, Mass. and I wish to thank them for their generous cooperation. The IBM system consisted of a 370 Mainframe, a 3705 Communications Processor, and a 3271 Terminal Controller. A Comlink II Modem supplied the modem interface and all communications took place at 4800 baud. In addition to observing correct responses, a Spectron D601B Datascope was used to verify the data exchanges. A block diagram of the system is shown in Figure 40. The actual verification was accomplished by the 8273 system receiving and responding to polls from the 3705. This method was used on both point-to-point and multi-point configurations. No attempt was made to implement any higher protocol software over that of the poll and poll responses since such software would not affect the verification of the 8273 implementation. As testimony to the ease of use of the 8273, the system worked on the first try.

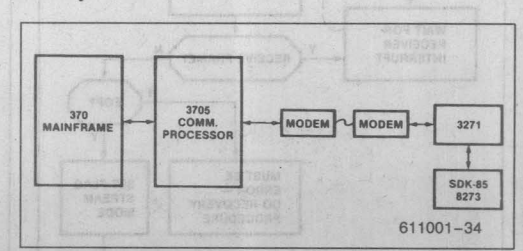


Figure 40. Raytheon Block Diagram

An SDK-85 (System Design Kit) was used as the core 8085 system. This system provides up to 4K bytes of ROM/EPROM, 512 bytes of RAM, 76 I/O pins, plus

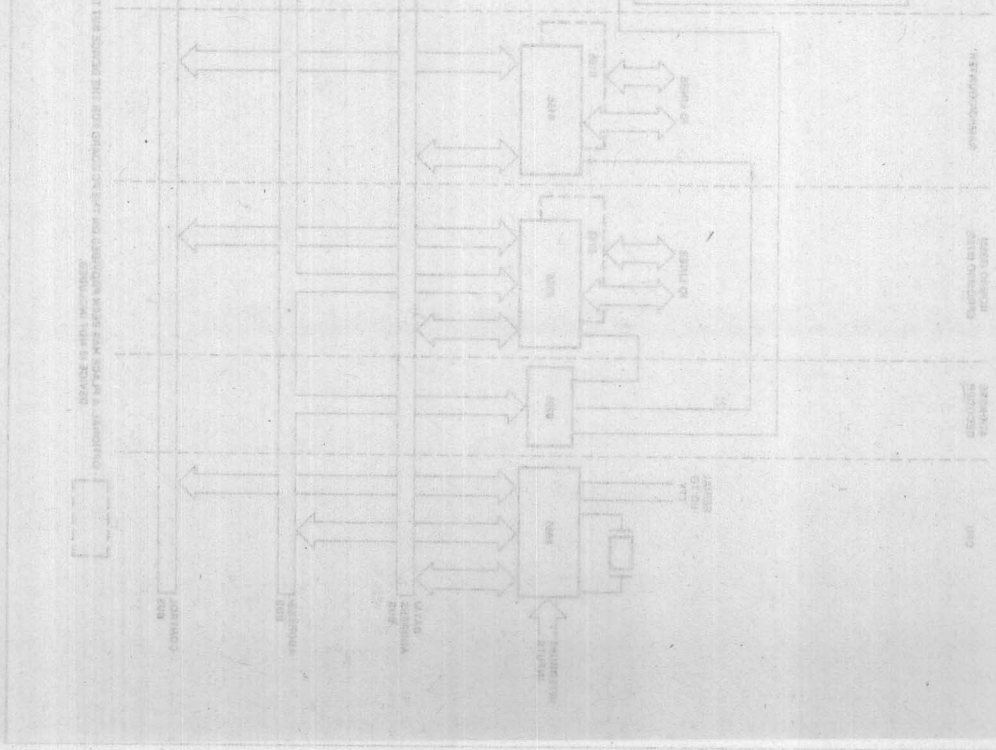
two timers as provided in two 8755 Combination EPROM/I/O devices and two 8155 Combination RAM/I/O/Timer devices. In addition, 5 interrupt inputs are supplied on the 8085. The address, data, and control buses are buffered by the 8212 and 8216 latches and bidirectional bus drivers. Although it was not used in this application, an 8279 Display Driver/Keyboard Encoder is included to interface the on-board display and keyboard. A block diagram of the SDK-85 is shown in Figure 41. The 8273 and associated circuitry was constructed on the ample wire-wrap area provided for the user.

The example 8237/8085 system is interrupt-driven and uses DMA for all data transfers supervised by an 8257 DMA Controller. A 2400 baud asynchronous line, implemented with an 8251A USART, provides communication between the software and the user. 8253 Programmable Interval Timer is used to supply the baud rate clocks for the 8251A and 8273. (The 8273 baud rate clocks were used only during initial system debug. In actual operation, the modem supplied these clocks via the RS-232 interface.) Two 2142 1K x 4 RAMs provided 512 bytes of transmitter and 512 bytes of receiver buffer memory. (Command and result buffers, plus miscellaneous variables are stored in the 8155s.) The RS-232 interface utilized MC1488 and MC1489 RS-232 drivers and receivers. The schematic of the system is shown in Figure 42.

One detail to note is the DMA and interrupt structure of the transmit and receive channels. In both cases, the receiver is always given the higher priority (8257 DMA channel 0 has priority over the remaining channels and the 8085 RST 7.5 interrupt input has priority over the RST 6.5 input.) Although the choice is arbitrary, this technique minimizes the chance that received data could be lost due to other processor or DMA commitments.

Also note that only one 8205 Decoder is used for both peripheral and memory Chip Select. This was done to eliminate separate memory and I/O decoders since it was known beforehand that neither address space would be completely filled.

The 4 MHz crystal and 8224 Clock Generator were used only to verify that the 8273 operates correctly at that maximum spec speed. In a normal system, the 3.072 MHz clock from the 8085 would be sufficient. (This fact was verified during initial checkout.)



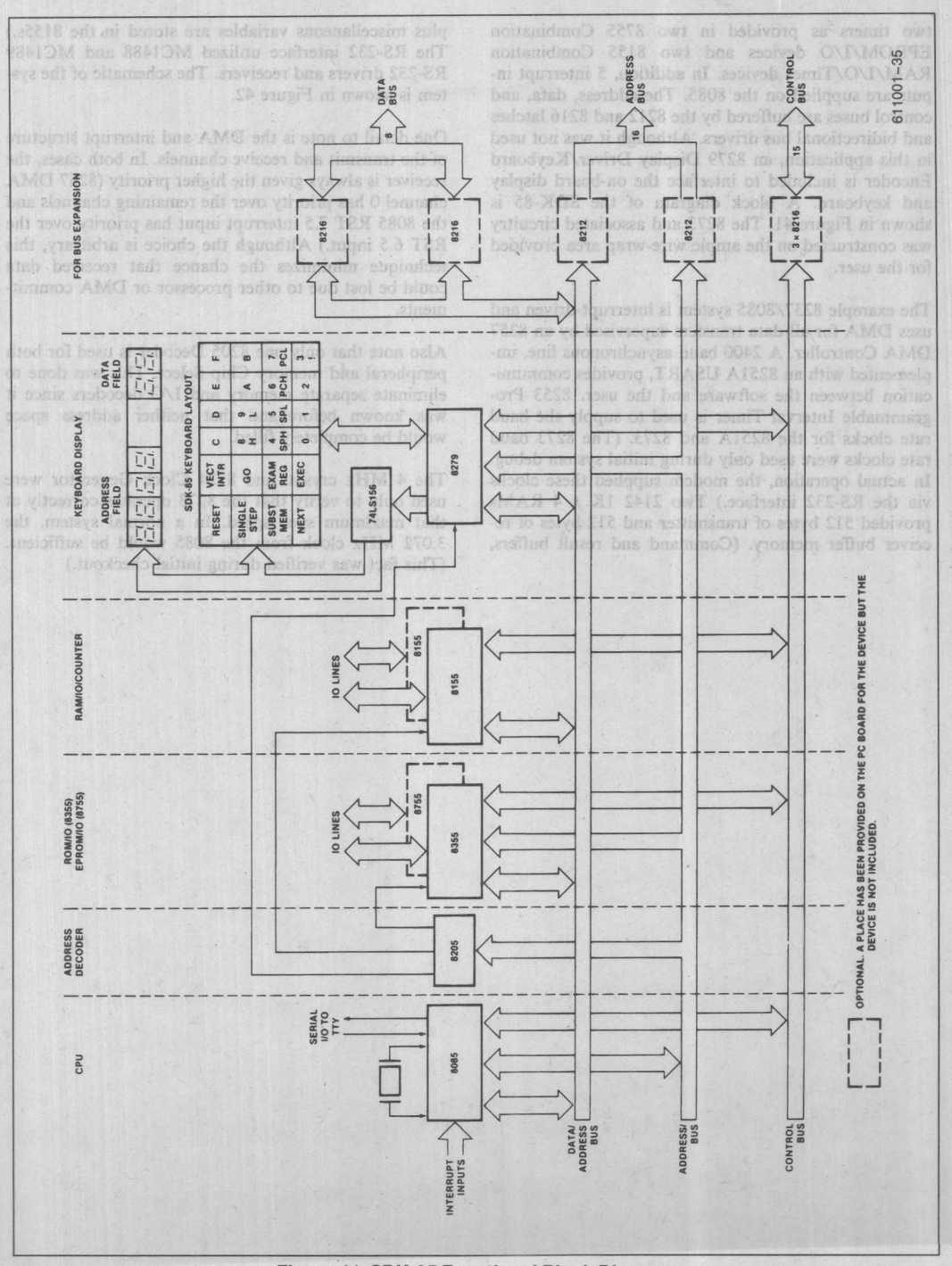


Figure 41. SDK-85 Functional Block Diagram

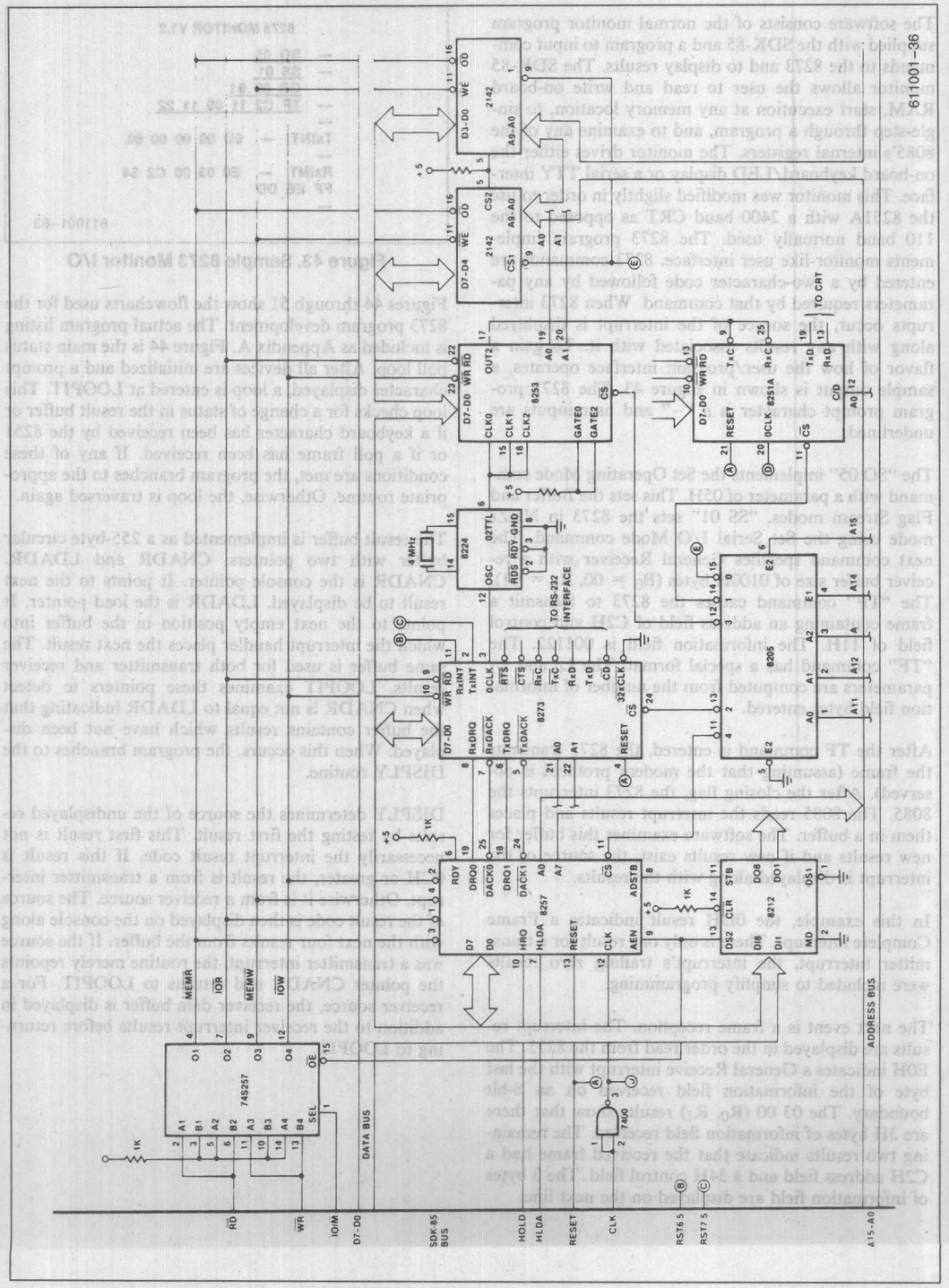


Figure 42. 8273/SDK-85 System

The software consists of the normal monitor program supplied with the SDK-85 and a program to input commands to the 8273 and to display results. The SDK-85 monitor allows the user to read and write on-board RAM, start execution at any memory location, to single-step through a program, and to examine any of the 8085's internal registers. The monitor drives either the on-board keyboard/LED display or a serial TTY interface. This monitor was modified slightly in order to use the 8251A with a 2400 baud CRT as opposed to the 110 baud normally used. The 8273 program implements monitor-like user interface. 8273 commands are entered by a two-character code followed by any parameters required by that command. When 8273 interrupts occur, the source of the interrupt is displayed along with any results associated with it. To gain a flavor of how the user/program interface operates, a sample output is shown in Figure 43. The 8273 program prompt character is a "–" and user inputs are underlined.

The "SO 05" implements the Set Operating Mode command with a parameter of 05H. This sets the Buffer and Flag Stream modes. "SS 01" sets the 8273 in NRZI mode using the Set Serial I/O Mode command. The next command specifies General Receiver with a receiver buffer size of 0100H bytes (B_0 = 00, B_1 = 01). The "TF" command causes the 8273 to transmit a frame containing an address field of C2H and control field of 11H. The information field is 001122. The "TF" command has a special format. The L_0 and L_1 parameters are computed from the number of information field bytes entered.

After the TF command is entered, the 8273 transmits the frame (assuming that the modem protocol is observed). After the closing flag, the 8273 interrupts the 8085. The 8085 reads the interrupt results and places them in a buffer. The software examines this buffer for new results and if new results exist, the source of the interrupt is displayed along with the results.

In this example, the 0DH result indicates a Frame Complete interrupt. There is only one result for a transmitter interrupt, the interrupt's trailing zero results were included to simplify programming.

The next event is a frame reception. The interrupt results are displayed in the order read from the 8273. The E0H indicates a General Receive interrupt with the last byte of the information field received on an 8-bit boundary. The 03 00 (R_0, R_1) results show that there are 3H bytes of information field received. The remaining two results indicate that the received frame had a C2H address field and a 34H control field. The 3 bytes of information field are displayed on the next line.

```
8273 MONITOR V1.2
— SO 05
— SS 01
— GR 00 01
— TF C2 11 00 11 22
—
TxINT — 0D 00 00 00 00
—
RxINT — E0 03 00 C2 34
FF EE DD
—
```

611001-63

Figure 43. Sample 8273 Monitor I/O

Figures 44 through 51 show the flowcharts used for the 8273 program development. The actual program listing is included as Appendix A. Figure 44 is the main status poll loop. After all devices are initialized and a prompt character displayed, a loop is entered at LOOPIT. This loop checks for a change of status in the result buffer or if a keyboard character has been received by the 8251 or if a poll frame has been received. If any of these conditions are met, the program branches to the appropriate routine. Otherwise, the loop is traversed again.

The result buffer is implemented as a 255-byte circular buffer with two pointers: CNADR and LDADR. CNADR is the console pointer. It points to the next result to be displayed. LDADR is the load pointer. It points to the next empty position in the buffer into which the interrupt handler places the next result. The same buffer is used for both transmitter and receiver results. LOOPIT examines these pointers to detect when CNADR is not equal to LDADR indicating that the buffer contains results which have not been displayed. When this occurs, the program branches to the DISPLY routine.

DISPLY determines the source of the undisplayed results by testing the first result. This first result is not necessarily the interrupt result code. If this result is 0CH or greater, the result is from a transmitter interrupt. Otherwise it is from a receiver source. The source of the result code is then displayed on the console along with the next four results from the buffer. If the source was a transmitter interrupt, the routine merely repoints the pointer CNADR and returns to LOOPIT. For a receiver source, the receiver data buffer is displayed in addition to the receiver interrupt results before returning to LOOPIT.

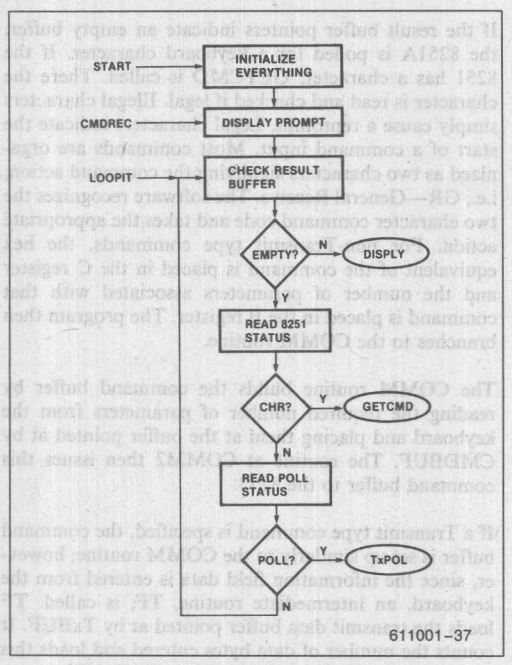


Figure 44. Main Status Poll Loop

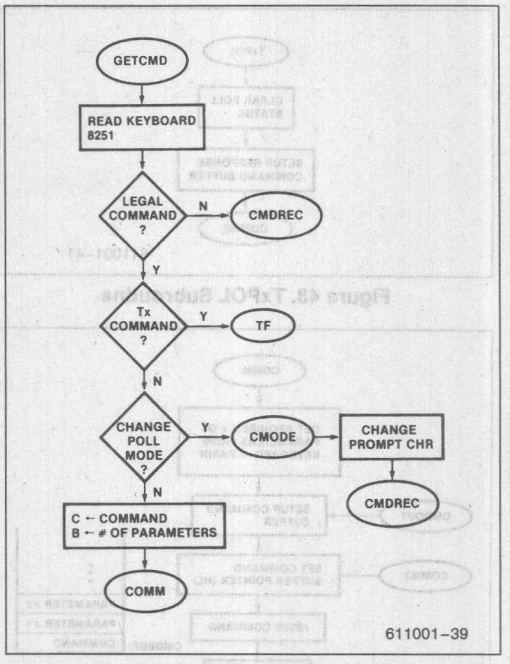


Figure 46. GETCMD Subroutine

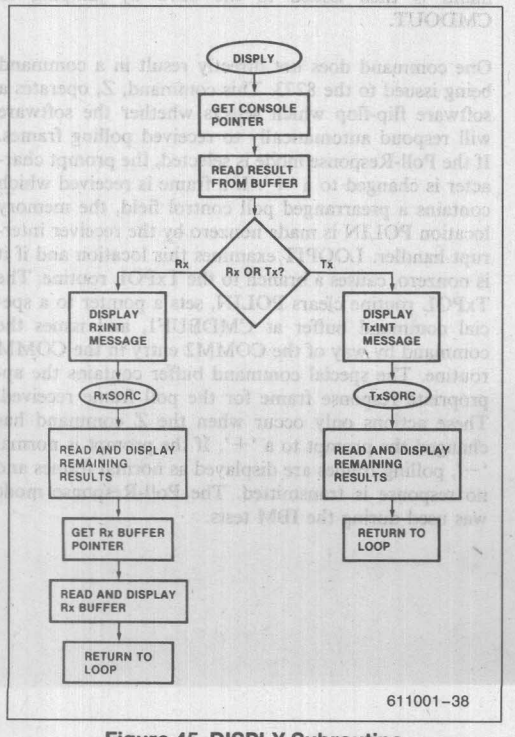


Figure 45. DISPLY Subroutine

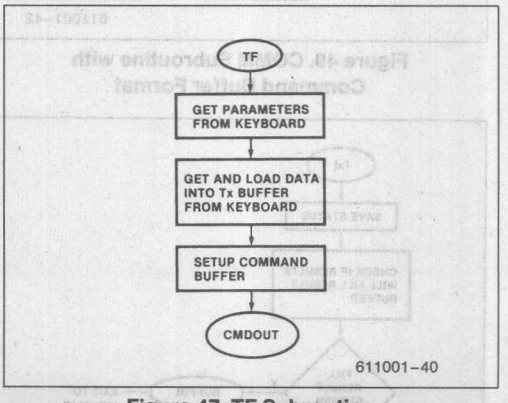


Figure 47. TF Subroutine

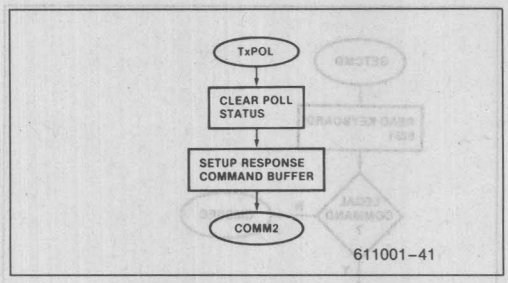


Figure 48. TxPOL Subroutine

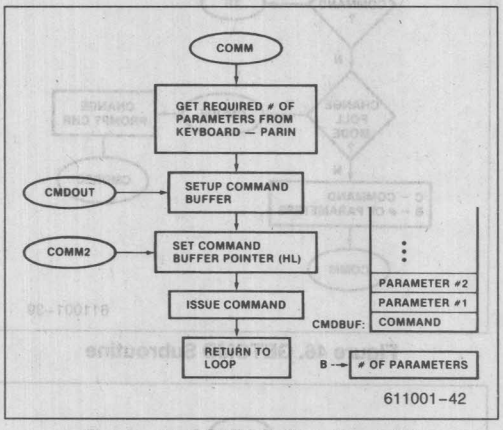


Figure 49. COMM Subroutine with Command Buffer Format

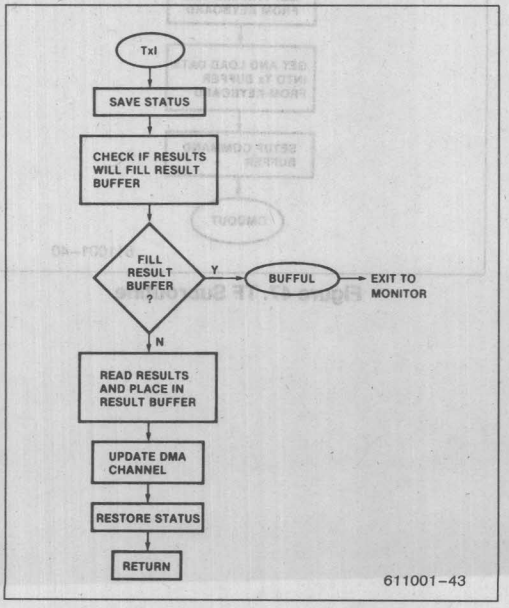


Figure 50. TxI (Transmitter Interrupt) Routine

If the result buffer pointers indicate an empty buffer, the 8251A is polled for a keyboard character. If the 8251 has a character, GETCMD is called. There the character is read and checked if legal. Illegal characters simply cause a reprompt. Legal characters indicate the start of a command input. Most commands are organized as two characters signifying the command action; i.e., GR—General Receive. The software recognizes the two character command code and takes the appropriate action. For non-Transmit type commands, the hex equivalent of the command is placed in the C register and the number of parameters associated with that command is placed in the B register. The program then branches to the COMM routine.

The COMM routine builds the command buffer by reading the required number of parameters from the keyboard and placing them at the buffer pointed at by CMDBUF. The routine at COMM2 then issues this command buffer to the 8273.

If a Transmit type command is specified, the command buffer is set up similarly to the COMM routine; however, since the information field data is entered from the keyboard, an intermediate routine, TF, is called. TF loads the transmit data buffer pointed at by TxBUF. It counts the number of data bytes entered and loads this number into the command buffer as L_0, L_1. The command is then issued to the 8273 by jumping to CMDOUT.

One command does not directly result in a command being issued to the 8273. This command, Z, operates a software flip-flop which selects whether the software will respond automatically to received polling frames. If the Poll-Response mode is selected, the prompt character is changed to a '+'. If a frame is received which contains a prearranged poll control field, the memory location POLIN is made nonzero by the receiver interrupt handler. LOOPIT examines this location and if it is nonzero, causes a branch to the TxPOL routine. The TxPOL routine clears POLIN, sets a pointer to a special command buffer at CMDBUF1, and issues the command by way of the COMM2 entry in the COMM routine. The special command buffer contains the appropriate response frame for the poll frame received. These actions only occur when the Z command has changed the prompt to a '+'. If the prompt is normal '−', polling frames are displayed as normal frames and no response is transmitted. The Poll-Response mode was used during the IBM tests.

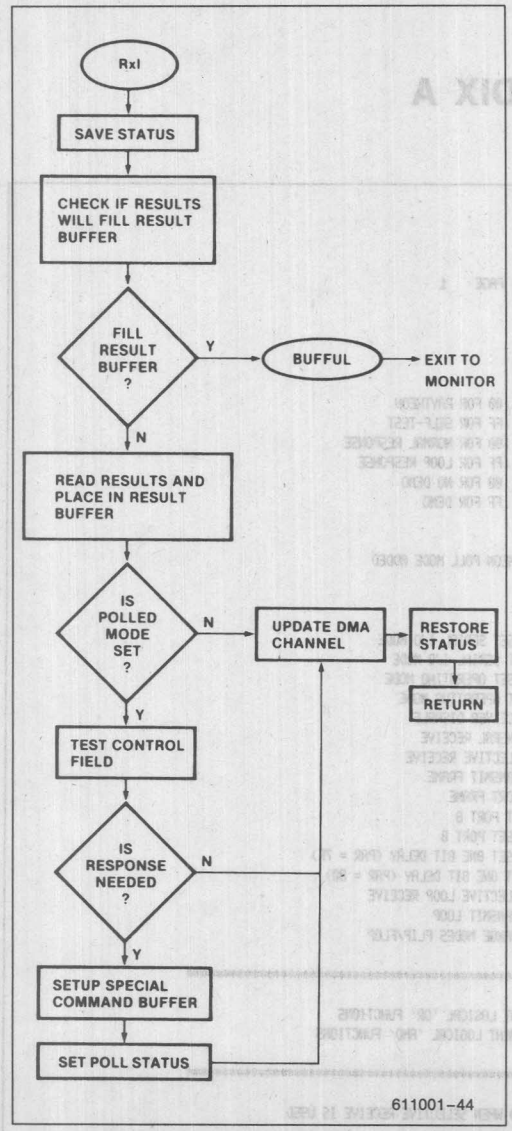


Figure 51. RxI (Receiver Interrupt) Routine

The final two software routines are the transmitter and receiver interrupt handlers. The transmit interrupt handler, TxI, simply saves the registers on the stack and checks if loading the result buffer will fill it. If the result buffer will overfill, the program is exited and control is passed to the SDK-85 monitor. If not, the results are read from the TxI/R register and placed in the result buffer at LDADR. The DMA pointers are then reset, the registers restored, and interrupts enabled. Execution then returns to the pre-interrupt location.

The receiver interrupt handler, RxI, is only slightly more complex. As in TxI, the registers are saved and the possibility of overfilling the result buffer is examined. If the result buffer is not full, the results are read from RxI/R and placed in the buffer. At this point the prompt character is examined to see if the Poll-Response mode is selected. If so, the control field is compared with two possible polling control fields. If there is a match, the special command buffer is loaded and the poll indicator, POLIN, is made nonzero. If no match occurred, no action is taken. Finally, the receiver DMA buffer pointers are reset, the processor status restored, and interrupts are enabled. The RET instruction returns execution to the pre-interrupt location.

This completes the discussion of the 8273/8085 system design.

CONCLUSION

This application note has covered the 8273 in some detail. The simple and low cost loop configuration was explored and an 8273/8085 system was presented as a sample design illustrating the DMA/interrupt-driven interface. It is hoped that the major features of the 8273, namely the frame-level command structure and the Digital Phase Locked Loop, have been shown to be a valuable asset in an SDLC system design.

APPENDIX A

```
ASM80 :F1:RAYT73.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, X108        MODULE   PAGE   1

 LOC  OBJ        SEQ        SOURCE STATEMENT

                   1 $NOPAGING MOD85  NOCOND
 0000              2 TRUE     EQU     00H          ;00 FOR RAYTHEON
                   3 ;                             ;FF FOR SELF-TEST
 0000              4 TRUE1    EQU     00H          ;00 FOR NORMAL RESPONSE
                   5 ;                             ;FF FOR LOOP RESPONSE
 0000              6 DEM      EQU     00H          ;00 FOR NO DEMO
                   7 ;                             ;FF FOR DEMO
                   8 ;
                   9 ;
                  10 ;GENERAL 8273 MONITOR WITH RAYTHEON POLL MODE ADDED
                  11 ;
                  17 ;
                  18 ;
                  19 ;COMMAND SUPPORTED ARE: RS - RESET SERIAL I/O MODE
                  20 ;                       SS - SET SERIAL I/O MODE
                  21 ;                       RO - RESET OPERATING MODE
                  22 ;                       SO - SET OPERATING MODE
                  23 ;                       RD - RECEIVER DISABLE
                  24 ;                       GR - GENERAL RECEIVE
                  25 ;                       SR - SELECTIVE RECEIVE
                  26 ;                       TF - TRANSMIT FRAME
                  27 ;                       AF - ABORT FRAME
                  28 ;                       SP - SET PORT B
                  29 ;                       RP - RESET PORT B
                  30 ;                       RB - RESET ONE BIT DELAY (PAR = 7F)
                  31 ;                       SB - SET ONE BIT DELAY (PAR = 80)
                  32 ;                       SL - SELECTIVE LOOP RECEIVE
                  33 ;                       TL - TRANSMIT LOOP
                  34 ;                       Z  - CHANGE MODES FLIP/FLOP
                  38 ;
                  39 ;*****************************************************************
                  40 ;
                  41 ;NOTE:  'SET' COMMANDS IMPLEMENT LOGICAL 'OR' FUNCTIONS
                  42 ;       'RESET' COMMANDS IMPLEMENT LOGICAL 'AND' FUNCTIONS
                  43 ;
                  44 ;*****************************************************************
                  45 ;
                  46 ;BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED.
                  47 ;
                  48 ;COMMAND FORMAT IS: 'COMMAND (2 LTRS)' 'PAR #1' 'PAR #2' ETC.
                  49 ;
                  50 ;THE TRANSMIT FRAME COMMAND FORMAT IS: 'TF' 'A' 'C' 'BUFFER CONTENTS'.
                  51 ;       NO LENGTH COUNT IS NEEDED.  BUFFER CONTENTS IS ENDED WITH A CR.
                  52 ;
                  53 ;*****************************************************************
                  54 ;
                  55 ;POLLED MODE:   WHEN POLLED MODE IS SELECTED (DENOTED BY A '+' PROMPT), IF
```

611001-45

```
                        56 ;            A SNRM-P OR RR(0)-P IS RECEIVED, A RESPONSE FRAME OF NSA-F
                        57 ;            OR RR(0)-F IS TRANSMITTED.  OTHER COMMANDS OPERATE NORMALLY.
                        62 ;
                        63 ;*************************************************************************
                        64 ;
                        65 ;8273 EQUATES
                        66 ;
0090                    67 STAT73  EQU   90H          ;STATUS REGISTER
0090                    68 COMM73  EQU   90H          ;COMMAND REGISTER
0091                    69 PARM73  EQU   91H          ;PARAMETER REGISTER
0091                    70 RESL73  EQU   91H          ;RESULT REGISTER
0092                    71 TXIR73  EQU   92H          ;TX INTERRUPT RESULT REGISTER
0093                    72 RXIR73  EQU   93H          ;RX INTERRUPT RESULT REGISTER
0092                    73 TEST73  EQU   92H          ;TEST MODE REGISTER
0020                    74 CPBF    EQU   20H          ;PARAMETER BUFFER FULL BIT
0004                    75 TXINT   EQU   04H          ;TX INTERRUPT BIT IN STATUS REGISTER
0008                    76 RXINT   EQU   08H          ;RX INTERRUPT BIT IN STATUS REGISTER
0001                    77 TXIRA   EQU   01H          ;TX INT RESULT AVAILABLE BIT
0002                    78 RXIRA   EQU   02H          ;RX INT RESULT AVAILABLE BIT
                        79 ;
                        80 ;8253 EQUATES
                        81 ;
009B                    82 MODE53  EQU   9BH          ;8253 MODE WORD REGISTER
009C                    83 CNT053  EQU   9CH          ;COUNTER 0 REGISTER
009D                    84 CNT153  EQU   9DH          ;COUNTER 1 REGISTER
009E                    85 CNT253  EQU   9EH          ;COUNTER 2 REGISTER
000C                    86 COBR    EQU   000CH        ;CONSOLE BAUD RATE (2400)
0036                    87 MDCNT0  EQU   36H          ;MODE FOR COUNTER 0
00B6                    88 MDCNT2  EQU   0B6H         ;MODE FOR COUNTER 2
2017                    89 LKBR1   EQU   2017H        ;8273 BAUD RATE LSB ADR
2018                    90 LKBR2   EQU   2018H        ;8273 BAUD RATE MSB ADR
                        91 ;
                        92 ;BAUD RATE TABLE:          BAUD RATE      LKBR1   LKBR2
                        93 ;                          *********      *****   *****
                        94 ;                           9600           2E      00
                        95 ;                           4800           5C      00
                        96 ;                           2400           B9      00
                        97 ;                           1200           72      01
                        98 ;                            600           E5      02
                        99 ;                            300           C9      05
                       100 ;
                       101 ;
                       102 ;8257 EQUATES
                       103 ;
00A8                   104 MODE57  EQU   0A8H         ;8257 MODE PORT
00A0                   105 CH0ADR  EQU   0A0H         ;CH0 (RX) ADR REGISTER
00A1                   106 CH0TC   EQU   0A1H         ;CH0 TERMINAL COUNT REGISTER
00A2                   107 CH1ADR  EQU   0A2H         ;CH1 (TX) ADR REGISTER
00A3                   108 CH1TC   EQU   0A3H         ;CH1 TERMINAL COUNT REGISTER
00A8                   109 STAT57  EQU   0A8H         ;STATUS REGISTER
8200                   110 RXBUF   EQU   8200H        ;RX BUFFER START ADDRESS
8000                   111 TXBUF   EQU   8000H        ;TX BUFFER START ADDRESS
0062                   112 DRDMA   EQU   62H          ;DISABLE RX DMA CHANNEL, TX STILL ON
41FF                   113 RXTC    EQU   41FFH        ;TERMINAL COUNT AND MODE FOR RX CHANNEL
0063                   114 ENDMA   EQU   63H          ;ENABLE BOTH TX AND RX CHANNELS-EXT. WR, TX STOP
0061                   115 DTDMA   EQU   61H          ;DISABLE TX DMA CHANNEL, RX STILL ON
81FF                   116 TXTC    EQU   81FFH        ;TERMINAL COUNT AND MODE FOR TX CHANNEL
                       117 ;
```

611001-46

2

```
                        118 ;8251A EQUATES
                        119 ;
0089                    120 CNTL51  EQU     89H             ;CONTROL WORD REGISTER
0089                    121 STAT51  EQU     89H             ;STATUS REGISTER
0088                    122 TXD51   EQU     88H             ;TX DATA REGISTER
0088                    123 RXD51   EQU     88H             ;RX DATA REGISTER
00CE                    124 MDE51   EQU     0CEH            ;MODE 16X, 2 STOP, NO PARITY
0027                    125 CMD51   EQU     27H             ;COMMAND, ENABLE TX&RX
0002                    126 RDY     EQU     02H             ;RXRDY BIT
                        127 ;
                        128 ;MONITOR SUBROUTINE EQUATES
                        129 ;
061F                    130 GETCH   EQU     061FH           ;GET CHR FROM KEYBOARD, ASCII IN CH
05F8                    131 ECHO    EQU     05F8H           ;ECHO CHR TO DISPLAY
075E                    132 VALDG   EQU     075EH           ;CHECK IF VALID DIGIT, CARRY SET IF VALID
05BB                    133 CNVBN   EQU     05BBH           ;CONVERTS ASCII TO HEX
05EB                    134 CRLF    EQU     05EBH           ;DISPLAY CR, HENCE LF TOO
06C7                    135 NMOUT   EQU     06C7H           ;CONVERT BYTE TO 2 ASCII CHR AND DISPLAY
                        136 ;
                        137 ;MISC EQUATES
                        138 ;
20C0                    139 STKSRT  EQU     20C0H           ;STACK START
0003                    140 CNTLC   EQU     03H             ;CNTL-C EQUIVALENT
0008                    141 MONTOR  EQU     0008H           ;MONITOR
2000                    142 CMDBUF  EQU     2000H           ;START OF COMMAND BUFFER
2020                    143 CMDBF1  EQU     2020H           ;POLL MODE SPECIAL TX COMMAND BUFFER
000D                    144 CR      EQU     0DH             ;ASCII CR
000A                    145 LF      EQU     0AH             ;ASCII LF
20D4                    146 RST75   EQU     20D4H           ;RST7.5 JUMP ADDRESS
20CE                    147 RST65   EQU     20CEH           ;RST6.5 JUMP ADDRESS
2010                    148 LDADR   EQU     2010H           ;RESULT BUFFER LOAD POINTER STORAGE
2013                    149 CNADR   EQU     2013H           ;RESULT BUFFER CONSOLE POINTER STORAGE
2800                    150 RESBUF  EQU     2800H           ;RESULT BUFFER START - 255 BYTES
0093                    151 SNRMP   EQU     93H             ;SNRM-P CONTROL CODE
0011                    152 RR0P    EQU     11H             ;RR(0)-P CONTROL CODE
0073                    153 NSAF    EQU     73H             ;NSA-F CONTROL CODE
0011                    154 RR0F    EQU     11H             ;RR(0)-F CONTROL CODE
2015                    155 PRMPT   EQU     2015H           ;PRMPT STORAGE
2016                    156 POLIN   EQU     2016H           ;POLL MODE SELECTION INDICATOR
2027                    157 DEMODE  EQU     2027H           ;DEMO MODE INDICATOR
                        161 ;
                        162 ;*************************************************************************
                        163 ;
                        164 ;RAM STORAGE DEFINITIONS:
                        165 ;       LOC             DEF
                        166 ;       ---             ---
                        167 ;       2000-200F       COMMAND BUFFER
                        168 ;       2010-2011       RESULT BUFFER LOAD POINTER
                        169 ;       2013-2014       RESULT BUFFER CONSOLE POINTER
                        170 ;       2015            PROMPT CHARACTER STORAGE
                        171 ;       2016            POLL MODE INDICATOR
                        172 ;       2017            BAUD RATE LSB FOR SELF-TEST
                        173 ;       2018            BAUD RATE MSB FOR SELF-TEST
                        177 ;       2019            SPARE
                        179 ;       2020-2026       RESPONSE COMMAND BUFFER FOR POLL MODE
                        180 ;       2800-28FF       RESULT BUFFER
                        181 ;
                        182 ;*************************************************************************
```

611001-47

```
                        183 ;
                        184 ;PROGRAM START
                        185 ;
                        186 ;INITIALIZE 8253, 8257, 8251A, AND RESET 8273.
                        187 ;ALSO SET NORMAL MODE, AND PRINT SIGNON MESSAGE
                        188 ;
0800                    189         ORG     800H
                        190
0800 31C820             191 START:  LXI     SP,STKSRT       ;INITIALIZE SP
0803 3E36               192         MVI     A,MDCNT0        ;8253 MODE SET
0805 D39B               193         OUT     MODE53          ;8253 MODE PORT
0807 3A1720             194         LDA     LKBR1           ;GET 8273 BAUD RATE LSB
080A D39C               195         OUT     CNT053          ;USING COUNTER 0 AS BAUD RATE GEN
080C 3A1820             196         LDA     LKBR2           ;GET 8273 BUAD RATE MSB
080F D39C               197         OUT     CNT053          ;COUNTER 0
0811 CD1A0B             198         CALL    RXDMA           ;INITIALIZE 8257 RX DMA CHANNEL
0814 CD350B             199         CALL    TXDMA           ;INITIALIZE 8257 TX DMA CHANNEL
0817 3E01               200         MVI     A,01H           ;OUTPUT 1 FOLLOWED BY A 0
0819 D392               201         OUT     TEST73          ;TO TEST MODE REGISTER
081B 3E00               202         MVI     A,00H           ;TO RESET THE 8273
081D D392               203         OUT     TEST73
081F 3E2D               204         MVI     A,'-'           ;NORMAL MODE PROMPT CHR
0821 321520             205         STA     PRMPT           ;PUT IN STORAGE
0824 3E00               206         MVI     A,00H           ;TX POLL RESPONSE INDICATOR
0826 321620             207         STA     POLIN           ;0 MEANS NO SPECIAL TX
0829 322720             208         STA     DEMODE          ;CLEAR DEMO MODE
082C 21A30C             212         LXI     H,SIGNON                ;SIGNON MESSAGE ADR
082F CD920C             213         CALL    TYMSG           ;DISPLAY SIGNON
                        214 ;
                        215 ;MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
                        216 ;
0832 21D420             217         LXI     H,RST75         ;RST7.5 JUMP LOCATION USED BY MONITOR
0835 01000C             218         LXI     B,RXI           ;ADDRESS OF RX INT ROUTINE
0838 36C3               219         MVI     M,0C3H          ;LOAD 'JMP' OPCODE
083A 23                 220         INX     H               ;INC POINTER
083B 71                 221         MOV     M,C             ;LOAD RXI LSB
083C 23                 222         INX     H               ;INC POINTER
083D 70                 223         MOV     M,B             ;LOAD RXI MSB
083E 21CE20             224         LXI     H,RST65         ;RST6.5 JUMP LOCATION USED BY MONITOR
0841 01CE0C             225         LXI     B,TXI           ;ADDRESS OF TX INT ROUTINE
0844 36C3               226         MVI     M,0C3H          ;LOAD 'JMP' OPCODE
0846 23                 227         INX     H               ;INC POINTER
0847 71                 228         MOV     M,C             ;LOAD TXI LSB
0848 23                 229         INX     H               ;INC POINTER
0849 70                 230         MOV     M,B             ;LOAD TXI MSB
084A 3E18               231         MVI     A,18H           ;GET SET TO RESET INTERRUPTS
084C 30                 232         SIM                     ;RESET INTERRUPTS
084D FB                 233         EI                      ;ENABLE INTERRUPTS
                        234 ;
                        235 ;INITIALIZE BUFFER POINTER
                        236 ;
                        237 ;
084E 210028             238         LXI     H,RESBUF        ;SET RESULT BUFFER POINTERS
0851 221320             239         SHLD    CNADR           ;RESULT CONSOLE POINTER
0854 221020             240         SHLD    LDADR           ;RESULT LOAD POINTER
                        241 ;
                        242 ;MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USART STATUS,
                        243 ;       OR POLL STATUS
```

611001-48

2

```
                        268 ;COMMAND RECOGNIZER ROUTINE
                        269 ;
                        270 ;
087D CD1F06             271 GETCMD: CALL    GETCH           ;GET CHR
0880 CDF805             272         CALL    ECHO            ;ECHO IT
0883 79                 273         MOV     A,C             ;SETUP FOR COMPARE
0884 FE52               274         CPI     'R'             ;R?
0886 CAAF08             275         JZ      RDWN            ;GET MORE
0889 FE53               276         CPI     'S'             ;S?
088B CAD708             277         JZ      SDWN            ;GET MORE
088E FE47               278         CPI     'G'             ;G?
0890 CAFF08             279         JZ      GDWN            ;GET MORE
0893 FE54               280         CPI     'T'             ;T?
0895 CA0E09             281         JZ      TDWN            ;GET MORE
0898 FE41               282         CPI     'A'             ;A?
089A CA2209             283         JZ      ADWN            ;GET MORE
089D FE5A               284         CPI     'Z'             ;Z?
089F CA3109             285         JZ      CMODE           ;YES, GO CHANGE MODE
08A2 FE03               290         CPI     CNTLC           ;CNTL-C?
08A4 CA0800             291         JZ      MONTOR          ;EXIT TO MONITOR
08A7 0E3F               292 ILLEG:  MVI     C,'?'           ;PRINT ?
08A9 CDF805             293         CALL    ECHO            ;DISPLAY IT
08AC C35708             294         JMP     CMDREC          ;LOOP FOR COMMAND
                        295
08AF CD1F06             296 RDWN:   CALL    GETCH           ;GET NEXT CHR
08B2 CDF805             297         CALL    ECHO            ;ECHO IT
08B5 79                 298         MOV     A,C             ;SETUP FOR COMPARE
08B6 FE4F               299         CPI     'O'             ;O?
08B8 CA5D09             300         JZ      ROCMD           ;RO COMMAND
08BB FE53               301         CPI     'S'             ;S?
08BD CA6709             302         JZ      RSCMD           ;RS COMMAND
08C0 FE44               303         CPI     'D'             ;D?
08C2 CA7109             304         JZ      RDCMD           ;RD COMMAND
08C5 FE50               305         CPI     'P'             ;P?
08C7 CAD809             306         JZ      RPCMD           ;RP COMMAND
08CA FE52               307         CPI     'R'             ;R?
08CC CA0008             308         JZ      START           ;START OVER
08CF FE42               309         CPI     'B'             ;B?
08D1 CA7B09             310         JZ      RBCMD           ;RB COMMAND
```

611001-49

```
08D4 C3A708      311         JMP     ILLEG           ;ILLEGAL, TRY AGAIN
                 312
08D7 CD1F06      313 SDWN:   CALL    GETCH           ;GET NEXT CHR
08DA CDF805      314         CALL    ECHO            ;ECHO IT
08DD 78          315         MOV     A,B             ;SETUP FOR COMPARE
08DE FE4F        316         CPI     'O'             ;O?
08E0 CAA609      317         JZ      SOCMD           ;SO COMMAND
08E3 FE53        318         CPI     'S'             ;S?
08E5 CAB009      319         JZ      SSCMD           ;SS COMMAND
08E8 FE52        320         CPI     'R'             ;R?
08EA CABA09      321         JZ      SRCMD           ;SR COMMAND
08ED FE50        322         CPI     'P'             ;P?
08EF CAE209      323         JZ      SPCMD           ;SP COMMAND
08F2 FE42        324         CPI     'B'             ;B?
08F4 CA8509      325         JZ      SBCMD           ;SB COMMAND
08F7 FE4C        326         CPI     'L'             ;L?
08F9 CA8F09      327         JZ      SLCMD           ;SL COMMAND
08FC C3A708      328         JMP     ILLEG           ;ILLEGAL, TRY AGAIN
                 329
08FF CD1F06      330 GDWN:   CALL    GETCH           ;GET NEXT CHR
0902 CDF805      331         CALL    ECHO            ;ECHO IT
0905 78          332         MOV     A,B             ;SETUP FOR COMPARE
0906 FE52        333         CPI     'R'             ;R?
0908 CAC409      334         JZ      GRCMD           ;GR COMMAND
090B C3A708      335         JMP     ILLEG           ;ILLEGAL, TRY AGAIN
                 336
090E CD1F06      337 TDWN:   CALL    GETCH           ;GET NEXT CHR
0911 CDF805      338         CALL    ECHO            ;ECHO IT
0914 78          339         MOV     A,B             ;SETUP FOR COMPARE
0915 FE46        340         CPI     'F'             ;F?
0917 CAEC09      341         JZ      TFCMD           ;TF COMMAND
091A FE4C        342         CPI     'L'             ;L?
091C CA9909      343         JZ      TLCMD           ;TL COMMAND
091F C3A708      344         JMP     ILLEG           ;ILLEGAL, TRY AGAIN
                 345
0922 CD1F06      346 ADWN:   CALL    GETCH           ;GET NEXT CHR
0925 CDF805      347         CALL    ECHO            ;ECHO IT
0928 78          348         MOV     A,B             ;SETUP FOR COMPARE
0929 FE46        349         CPI     'F'             ;F?
092B CACE09      350         JZ      AFCMD           ;AF COMMAND
092E C3A708      351         JMP     ILLEG           ;ILLEGAL, TRY AGAIN
                 352 ;
                 353 ;RESET POLL MODE RESPONSE - CHANGE PROMPT CHR AS INDICATOR
                 354 ;
0931 F3          355 CMODE:  DI                      ;DISABLE INTERRUPTS
0932 3A1520      356         LDA     PRMPT           ;GET CURRENT PROMPT
0935 FE2D        357         CPI     '-'             ;NORMAL MODE?
0937 C24309      358         JNZ     SW              ;NO, CHANGE IT
093A 3E2B        359         MVI     A,'+'           ;NEW PROMPT
093C 321520      360         STA     PRMPT           ;STORE NEW PROMPT
093F FB          365         EI                      ;ENABLE INTERRUPTS
0940 C35708      366         JMP     CMDREC          ;RETURN TO LOOP
0943 3E2D        367 SW:     MVI     A,'-'           ;NEW PROMPT CHR
0945 321520      368         STA     PRMPT           ;STORE IT
0948 FB          369         EI                      ;ENABLE INTERRUPTS
0949 C35708      370         JMP     CMDREC          ;RETURN TO LOOP
                 371 ;
                 372 ;
```

611001-50

```
                        373 ;TRANSMIT ANSWER TO POLL SETUP
                        374 ;
094C 3E00               382 TXPOL:  MVI     A,00H           ;CLEAR POLL INDICATOR
094E 321620             384         STA     POLIN           ;INDICATOR ADR
0951 216108             385         LXI     H,LOOPIT        ;SETUP STACK FOR COMMAND OUTPUT
0954 E5                 386         PUSH    H               ;PUT RETURN TO CMDREC ON STACK
0955 0604               387         MVI     B,04H           ;GET # OF PARAMETERS READY
0957 212020             388         LXI     H,CMDBF1        ;POINT TO SPECIAL BUFFER
095A C3FF0A             389         JMP     COMM2           ;JUMP TO COMMAND OUTPUTER
                        390 ;
                        391 ;
                        392 ;
                        393 ;COMMAND IMPLEMENTING ROUTINES
                        394 ;
                        395 ;
                        396 ;RO - RESET OPERATING MODE
                        397 ;
095D 0601               398 ROCMD:  MVI     B,01H           ;# OF PARAMETERS
095F 0E51               399         MVI     C,51H           ;COMMAND
0961 CDE50A             400         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0964 C35708             401         JMP     CMDREC          ;GET NEXT COMMAND
                        402 ;
                        403 ;RS - RESET SERIAL I/O MODE COMMAND
                        404 ;
0967 0601               405 RSCMD:  MVI     B,01H           ;# OF PARAMETERS
0969 0E60               406         MVI     C,60H           ;COMMAND
096B CDE50A             407         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
096E C35708             408         JMP     CMDREC          ;GET NEXT COMMAND
                        409 ;
                        410 ;RD - RECEIVER DISABLE COMMAND
                        411 ;
0971 0600               412 RDCMD:  MVI     B,00H           ;# OF PARAMETERS
0973 0EC5               413         MVI     C,0C5H          ;COMMAND
0975 CDE50A             414         CALL    COMM            ;ISSUE COMMAND
0978 C35708             415         JMP     CMDREC          ;GET NEXT COMMAND
                        416 ;
                        417 ;RB - RESET ONE BIT DELAY COMMAND
                        418 ;
097B 0601               419 RBCMD:  MVI     B,01H           ;# OF PARAMETERS
097D 0E64               420         MVI     C,64H           ;COMMAND
097F CDE50A             421         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
0982 C35708             422         JMP     CMDREC          ;GET NEXT COMMAND
                        423 ;
                        424 ;SB - SET ONE BIT DELAY COMMAND
                        425 ;
0985 0601               426 SBCMD:  MVI     B,01H           ;# OF PARAMETERS
0987 0EA4               427         MVI     C,0A4H          ;COMMAND
0989 CDE50A             428         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
098C C35708             429         JMP     CMDREC          ;GET NEXT COMMAND
                        430 ;
                        431 ;SL - SELECTIVE LOOP RECEIVE COMMAND
                        432 ;
098F 0604               433 SLCMD:  MVI     B,04H           ;# OF PARAMETERES
0991 0EC2               434         MVI     C,0C2H          ;COMMAND
0993 CDE50A             435         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
0996 C35708             436         JMP     CMDREC          ;GET NEXT COMMAND
                        437 ;
                        438 ;TL - TRANSMIT LOOP COMMAND
```

611001-51

```
                        439 ;
0999 210020             440 TLCMD:  LXI     H,CMDBUF        ;SET COMMAND BUFFER POINTER
099C 0602               441         MVI     B,02H           ;LOAD PARAMETER COUNTER
099E 36CA               442         MVI     M,0CAH          ;LOAD COMMAND INTO BUFFER
09A0 210220             443         LXI     H,CMDBUF+2      ;POINT AT ADR AND CNTL POSITIONS
09A3 C3F609             444         JMP     TFCMD1          ;FINISH OFF COMMAND IN TF ROUTINE
                        445 ;
                        446 ;SO - SET OPERATING MODE COMMAND
                        447 ;
09A6 0601               448 SOCMD:  MVI     B,01H           ;# OF PARAMETERS
09A8 0E91               449         MVI     C,91H           ;COMMAND
09AA CDE50A             450         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09AD C35708             451         JMP     CMDREC          ;GET NEXT COMMAND
                        452 ;
                        453 ;SS - SET SERIAL I/O COMMAND
                        454 ;
09B0 0601               455 SSCMD:  MVI     B,01H           ;# OF PARAMETERS
09B2 0EA0               456         MVI     C,0A0H          ;COMMAND
09B4 CDE50A             457         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09B7 C35708             458         JMP     CMDREC          ;GET NEXT COMMAND
                        459 ;
                        460 ;SR - SELECTIVE RECEIVE COMMAND
                        461 ;
09BA 0604               462 SRCMD:  MVI     B,04H           ;# OF PARAMETERS
09BC 0EC1               463         MVI     C,0C1H          ;COMMAND
09BE CDE50A             464         CALL    COMM            ;GET PARAMETERS AND ISSUE COMMAND
09C1 C35708             465         JMP     CMDREC          ;GET NEXT COMMAND
                        466 ;
                        467 ;GR - GENERAL RECEIVE COMMAND
                        468 ;
09C4 0602               469 GRCMD:  MVI     B,02H           ;NO PARAMETERS
09C6 0EC0               470         MVI     C,0C0H          ;COMMAND
09C8 CDE50A             471         CALL    COMM            ;ISSUE COMMAND
09CB C35708             472         JMP     CMDREC          ;GET NEXT COMMAND
                        473 ;
                        474 ;AF - ABORT FRAME COMMAND
                        475 ;
09CE 0600               476 AFCMD:  MVI     B,00H           ;NO PARAMETERS
09D0 0ECC               477         MVI     C,0CCH          ;COMMAND
09D2 CDE50A             478         CALL    COMM            ;ISSUE COMMAND
09D5 C35708             479         JMP     CMDREC          ;GET NEXT COMMAND
                        480 ;
                        481 ;RP - RESET PORT COMMAND
                        482 ;
09D8 0601               483 RPCMD:  MVI     B,01H           ;# OF PARAMETERS
09DA 0E63               484         MVI     C,63H           ;COMMAND
09DC CDE50A             485         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09DF C35708             486         JMP     CMDREC          ;GET NEXT COMMAND
                        487 ;
                        488 ;SP - SET PORT COMMAND
                        489 ;
09E2 0601               490 SPCMD:  MVI     B,01H           ;# OF PARAMETERS
09E4 0EA3               491         MVI     C,0A3H          ;COMMAND
09E6 CDE50A             492         CALL    COMM            ;GET PARAMETER AND ISSUE COMMAND
09E9 C35708             493         JMP     CMDREC          ;GET NEX COMMAND
                        494 ;
                        495 ;TF - TRANSMIT FRAME COMMAND
                        496 ;
```

611001-52

2

```
09EC 210020      497 TFCMD:  LXI    H,CMDBUF        ;SET COMMAND BUFFER POINTER
09EF 0602        498         MVI    B,02H           ;LOAD PARAMETER COUNTER
09F1 36C8        499         MVI    M,0C8H          ;LOAD COMMAND INTO BUFFER
09F3 210220      500         LXI    H,CMDBUF+2      ;POINT AT ADR AND CNTL POSITIONS
09F6 78          501 TFCMD1: MOV    A,B             ;TEST PARAMETER COUNT
09F7 A7          502         ANA    A               ;IS IT 0?
09F8 CA070A      503         JZ     TBUFL           ;YES, LOAD TX DATA BUFFER
09FB CDAD0A      504         CALL   PARIN           ;GET PARAMETER
09FE DAA708      505         JC     ILLEG           ;ILLEGAL CHR RETURNED
0A01 23          506         INX    H               ;INC COMMAND BUFFER POINTER
0A02 05          507         DCR    B               ;DEC PARAMETER COUNTER
0A03 77          508         MOV    M,A             ;LOAD PARAMETER INTO COMMAND BUFFER
0A04 C3F609      509         JMP    TFCMD1          ;GET NEXT PARAMETER
                 510
0A07 210080      511 TBUFL:  LXI    H,TXBUF         ;LOAD TX DATA BUFFER POINTER
0A0A 010000      512         LXI    B,0000H         ;CLEAR BC - BYTE COUNTER
0A0D C5          513 TBUFL1: PUSH   B               ;SAVE BYTE COUNTER
0A0E CDAD0A      514         CALL   PARIN           ;GET DATA, ALIAS PARAMETER
0A11 DA1B0A      515         JC     ENDCHK          ;MAYBE END IF ILLEGAL
0A14 77          516         MOV    M,A             ;LOAD DATA BYTE INTO BUFFER
0A15 23          517         INX    H               ;INC BUFFER POINTER
0A16 C1          518         POP    B               ;RESTORE BYTE COUNTER
0A17 03          519         INX    B               ;INC BYTE COUNTER
0A18 C30D0A      520         JMP    TBUFL1          ;GET NEXT DATA
0A1B FE0D        521 ENDCHK: CPI    CR              ;RETURNED ILLEGAL CHR CR?
0A1D CA240A      522         JZ     TBUFFL          ;YES, THEN TX BUFFER FULL
0A20 C1          523         POP    B               ;RESTORE B TO SAVE STACK
0A21 C3A708      524         JMP    ILLEG           ;ILLEGAL CHR
0A24 C1          525 TBUFFL: POP    B               ;RESTORE BYTE COUNTER
0A25 210120      526         LXI    H,CMDBUF+1      ;POINT INTO COMMAND BUFFER
0A28 71          527         MOV    M,C             ;STORE BYTE COUNT LSB
0A29 23          528         INX    H               ;INC POINTER
0A2A 70          529         MOV    M,B             ;STORE BYTE COUNT MSB
0A2B 0604        530         MVI    B,04H           ;LOAD PARAMETER COUNT INTO B
0A2D 21360A      531         LXI    H,TFRET         ;GET RETURN ADR FOR THIS ROUTINE
0A30 C5          532         PUSH   B               ;PUSH ONCE
0A31 E3          533         XTHL                   ;PUT RETURN ON STACK
0A32 C5          534         PUSH   B               ;PUSH IT SO CMDOUT CAN USE IT
0A33 C3FB0A      535         JMP    CMDOUT          ;ISSUE COMMAND
0A36 C35708      536 TFRET:  JMP    CMDREC          ;GET NEXT COMMAND
                 537 ;
                 538 ;
                 539 ;ROUTINE TO DISPLAY RESULT IN RESULT BUFFER WHEN LOAD AND CONSOLE
                 540 ;POINTERS ARE DIFFERENT.
                 541 ;
                 542 ;
0A39 1605        543 DISPY:  MVI    D,05H           ;D IS RESULT COUNTER
0A3B 2A1320      544         LHLD   CNADR           ;GET CONSOLE POINTER
0A3E E5          545         PUSH   H               ;SAVE IT
0A3F 7E          546         MOV    A,M             ;GET RESULT IC
0A40 E61F        547         ANI    1FH             ;LIMIT TO RESULT CODE
0A42 FE0C        548         CPI    0CH             ;TEST IF RX OR TX SOURCE
0A44 DA620A      549         JC     RXSORC          ;CARRY, THEN RX SOURCE
0A47 21C30C      550 TXSORC: LXI    H,TXIMSG        ;TX INT MESSAGE
0A4A CD920C      551         CALL   TYMSG           ;DISPLAY IT
0A4D E1          552 DISPY2: POP    H               ;RESTORE CONSOLE POINTER
0A4E 7E          553 DISPY1: MOV    A,M             ;GET RESULT
0A4F CDC706      554         CALL   NMOUT           ;CONVERT AND DISPLAY
```

611001-53

```
0A52 0E20       555         MVI     C,' '           ;SP CHR
0A54 CDF805     556         CALL    ECHO            ;DISPLAY IT
0A57 2C         557         INR     L               ;INC BUFFER POINTER
0A58 15         558         DCR     D               ;DEC RESULT COUNTER
0A59 C24E0A     559         JNZ     DISPY1          ;NOT DONE
0A5C 221320     560         SHLD    CNADR           ;UPDATE CONSOLE POINTER
0A5F C35708     561         JMP     CMDREC          ;RETURN TO LOOP
                562 ;
                563 ;
                564 ;RECEIVER SOURCE - DISPLAY RESULTS AND RECEVIE BUFFER CONTENTS
                565 ;
                566 ;
0A62 21B80C     567 RXSORC: LXI     H,RXIMSG        ;RX INT MESSAGE ADR
0A65 CD920C     568         CALL    TYMSG           ;DISPLAY MESSAGE
0A68 E1         569         POP     H               ;RESTORE CONSOLE POINTER
0A69 7E         570 RXS1:   MOV     A,M             ;RETRIEVE RESULT FROM BUFFER
0A6A CDC706     571         CALL    NMOUT           ;CONVERT AND DISPLAY IT
0A6D 0E20       572         MVI     C,' '           ;ASCII SP
0A6F CDF805     573         CALL    ECHO            ;DISPLAY IT
0A72 2C         574         INR     L               ;INC CONSOLE POINTER
0A73 15         575         DCR     D               ;DEC RESULT COUNTER
0A74 7A         576         MOV     A,D             ;GET SET TO TEST COUNTER
0A75 FE04       577         CPI     04H             ;IS THE RESULT R0?
0A77 CAA20A     578         JZ      R0PT            ;YES, GO SAVE IT
0A7A FE03       579         CPI     03H             ;IS THE RESULT R1?
0A7C CAA70A     580         JZ      R1PT            ;YES, GO SAVE IT
0A7F A7         581 RXS2:   ANA     A               ;TEST RESULT COUNTER
0A80 C2690A     582         JNZ     RXS1            ;NOT DONE YET, GET NEXT RESULT
0A83 221320     583         SHLD    CNADR           ;DONE, SO UPDATE CONSOLE POINTER
0A86 CDEB05     584         CALL    CRLF            ;DISPLAY CR
0A89 210082     585         LXI     H,RXBUF         ;POINT AT RX BUFFER
0A8C C1         586         POP     B               ;RETRIEVE RECEIVED COUNT
0A8D 78         587 RXS3:   MOV     A,B             ;IS COUNT 0?
0A8E B1         588         ORA     C               ;
0A8F CA5708     589         JZ      CMDREC          ;YES, GO BACK TO LOOP
0A92 7E         590         MOV     A,M             ;NO, GET CHR
0A93 C5         591         PUSH    B               ;SAVE BC
0A94 CDC706     592         CALL    NMOUT           ;CONVERT AND DISPLAY CHR
0A97 0E20       593         MVI     C,' '           ;ASCII SP
0A99 CDF805     594         CALL    ECHO            ;DISPLAY IT TO SEPARATE DATA
0A9C C1         595         POP     B               ;RESTORE BC
0A9D 0B         596         DCX     B               ;DEC COUNT
0A9E 23         597         INX     H               ;INC POINTER
0A9F C38D0A     598         JMP     RXS3            ;GET NEXT CHR
                599
0AA2 4E         600 R0PT:   MOV     C,M             ;GET R0 FOR RESULT BUFFER
0AA3 C5         601         PUSH    B               ;SAVE IT
0AA4 C37F0A     602         JMP     RXS2            ;RETURN
                603
0AA7 C1         604 R1PT:   POP     B               ;GET R0
0AA8 46         605         MOV     B,M             ;GET R1 FOR RESULT BUFFER
0AA9 C5         606         PUSH    B               ;SAVE IT
0AAA C37F0A     607         JMP     RXS2
                608 ;
                609 ;
                610 ;
                611 ;PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER
                612 ;
```

611001-54

```
                        613 ;
0AAD C5                 614 PARIN:  PUSH    B           ;SAVE BC
0AAE 1601               615         MVI     D,01H       ;SET CHR COUNTER
0AB0 CD1F06             616         CALL    GETCH       ;GET CHR
0AB3 CDF805             617         CALL    ECHO        ;ECHO IT
0AB6 79                 618         MOV     A,C         ;PUT CHR IN A
0AB7 FE20               619         CPI     ' '         ;SP?
0AB9 C2E00A             620         JNZ     PARIN1      ;NO, ILLEGAL, TRY AGAIN
0ABC CD1F06             621 PARIN3: CALL    GETCH       ;GET CHR OF PARAMETER
0ABF CDF805             622         CALL    ECHO        ;ECHO IT
0AC2 CD5E07             623         CALL    VALDG       ;IS IT A VALID CHR?
0AC5 D2E00A             624         JNC     PARIN1      ;NO, TRY AGAIN
0AC8 CDBB05             625         CALL    CNVBN       ;CONVERT IT TO HEX
0ACB 4F                 626         MOV     C,A         ;SAVE IT IN C
0ACC 7A                 627         MOV     A,D         ;GET CHR COUNTER
0ACD A7                 628         ANA     A           ;IS IT 0?
0ACE CADC0A             629         JZ      PARIN2      ;YES, DONE WITH THIS PARAMETER
0AD1 15                 630         DCR     D           ;DEC CHR COUNTER
0AD2 AF                 631         XRA     A           ;CLEAR CARRY
0AD3 79                 632         MOV     A,C         ;RECOVER 1ST CHR
0AD4 17                 633         RAL                 ;ROTATE LEFT 4 PLACES
0AD5 17                 634         RAL
0AD6 17                 635         RAL
0AD7 17                 636         RAL
0AD8 5F                 637         MOV     E,A         ;SAVE IT IN E
0AD9 C3BC0A             638         JMP     PARIN3      ;GET NEXT CHR
0ADC 79                 639 PARIN2: MOV     A,C         ;2ND CHR IN A
0ADD B3                 640         ORA     E           ;COMBINE BOTH CHRS
0ADE C1                 641         POP     B           ;RESTORE BC
0ADF C9                 642         RET                 ;RETURN TO CALLING PROGRAM
0AE0 79                 643 PARIN1: MOV     A,C         ;PUT ILLEGAL CHR IN A
0AE1 37                 644         STC                 ;SET CARRY AS ILLEGAL STATUS
0AE2 C1                 645         POP     B           ;RESTORE BC
0AE3 C9                 646         RET                 ;RETURN TO CALLING PROGRAM
                        647 ;
                        648 ;
                        649 ;JUMP HERE IF BUFFER FULL
                        650 ;
0AE4 CF                 651 BUFFUL: DB      0CFH        ;EXIT TO MONITOR
                        652 ;
                        653 ;
                        654 ;COMMAND DISPATCHER
                        655 ;
                        656 ;
0AE5 210020             657 COMM:   LXI     H,CMDBUF    ;SET POINTER
0AE8 C5                 658         PUSH    B           ;SAVE BC
0AE9 71                 659         MOV     M,C         ;LOAD COMMAND INTO BUFFER
0AEA 78                 660 COMM1:  MOV     A,B         ;CHECK PARAMETER COUNTER
0AEB A7                 661         ANA     A           ;IS IT 0?
0AEC CAFB0A             662         JZ      CMDOUT      ;YES, GO ISSUE COMMAND
0AEF CDAD0A             663         CALL    PARIN       ;GET PARAMETER
0AF2 DAA708             664         JC      ILLEG       ;ILLEGAL CHR RETURNED
0AF5 23                 665         INX     H           ;INC BUFFER POINTER
0AF6 05                 666         DCR     B           ;DEC PARAMETER COUNTER
0AF7 77                 667         MOV     M,A         ;PARAMETER TO BUFFER
0AF8 C3EA0A             668         JMP     COMM1       ;GET NEXT PARAMETER
0AFB 210020             669 CMDOUT: LXI     H,CMDBUF    ;REPOINT POINTER
0AFE C1                 670         POP     B           ;RESTORE PARAMETER COUNT
```

611001-55

```
0AFF DB90       671 COMM2:  IN      STAT73          ;READ 8273 STATUS
0B01 07         672         RLC                     ;ROTATE CBSY INTO CARRY
0B02 DAFF0A     673         JC      COMM2           ;WAIT FOR OK
0B05 7E         674         MOV     A,M             ;OK, MOVE COMMAND INTO A
0B06 D390       675         OUT     COMM73          ;OUTPUT COMMAND
0B08 78         676 PAR1:   MOV     A,B             ;GET PARAMETER COUNT
0B09 A7         677         ANA     A               ;IS IT 0?
0B0A C8         678         RZ                      ;YES, DONE, RETURN
0B0B 23         679         INX     H               ;INC COMMAND BUFFER POINTER
0B0C 05         680         DCR     B               ;DEC PARAMETER COUNT
0B0D DB90       681 PAR2:   IN      STAT73          ;READ STATUS
0B0F E620       682         ANI     CPBF            ;IS CPBF BIT SET?
0B11 C20D0B     683         JNZ     PAR2            ;WAIT TIL ITS 0
0B14 7E         684         MOV     A,M             ;OK, GET PARAMETER FROM BUFFER
0B15 D391       685         OUT     PARM73          ;OUTPUT PARAMETER
0B17 C3080B     686         JMP     PAR1            ;GET NEXT PARAMETER
                687 ;
                688 ;
                689 ;INITIALIZE AND ENABLE RX DMA CHANNEL
                690 ;
                691 ;
0B1A 3E62       692 RXDMA:  MVI     A,DRDMA         ;DISABLE RX DMA CHANNEL
0B1C D3A8       693         OUT     MODE57          ;8257 MODE PORT
0B1E 010002     694         LXI     B,RXBUF         ;RX BUFFER START ADDRESS
0B21 79         695         MOV     A,C             ;RX BUFFER LSB
0B22 D3A0       696         OUT     CH0ADR          ;CH0 ADR PORT
0B24 78         697         MOV     A,B             ;RX BUFFER MSB
0B25 D3A0       698         OUT     CH0ADR          ;CH0 ADR PORT
0B27 01FF41     699         LXI     B,RXTC          ;RX CH TEERMINAL COUNT
0B2A 79         700         MOV     A,C             ;RX TERMINAL COUNT LSB
0B2B D3A1       701         OUT     CH0TC           ;CH0 TC PORT
0B2D 78         702         MOV     A,B             ;RX TERMINAL COUNT MSB
0B2E D3A1       703         OUT     CH0TC           ;CH0 TC PORT
0B30 3E63       704         MVI     A,ENDMA         ;ENABLE DMA WORD
0B32 D3A8       705         OUT     MODE57          ;8257 MODE PORT
0B34 C9         706         RET                     ;RETURN
                707 ;
                708 ;
                709 ;INITIALIZE AND ENABLE TX DMA CHANNEL
                710 ;
                711 ;
0B35 3E61       712 TXDMA:  MVI     A,DTDMA         ;DISABLE TX DMA CHANNEL
0B37 D3A8       713         OUT     MODE57          ;8257 MODE PORT
0B39 010080     714         LXI     B,TXBUF         ;TX BUFFER START ADDRESS
0B3C 79         715         MOV     A,C             ;TX BUFFER LSB
0B3D D3A2       716         OUT     CH1ADR          ;CH1 ADR PORT
0B3F 78         717         MOV     A,B             ;TX BUFFER MSB
0B40 D3A2       718         OUT     CH1ADR          ;CH1 ADR PORT
0B42 01FF81     719 TXDMA1: LXI     B,TXTC          ;TX CH TERMINAL COUNT
0B45 79         720         MOV     A,C             ;TX TERMINAL COUNT LSB
0B46 D3A3       721         OUT     CH1TC           ;CH1 TC PORT
0B48 78         722         MOV     A,B             ;TX TERMINAL COUNT MSB
0B49 D3A3       723         OUT     CH1TC           ;CH1 TC PORT
0B4B 3E63       724         MVI     A,ENDMA         ;ENABLE DMA WORD
0B4D D3A8       725         OUT     MODE57          ;8257 MODE PORT
0B4F C9         726         RET                     ;RETURN
                727 ;
                728 ;
```

611001-56

2

```
                        729 ;INERRUPT PROCESSING SECTION
                        730 ;
0C00                    731         ORG     0C00H
                        732 ;
                        733 ;
                        734 ;RECEIVER INTERRUPT - RST 7.5 (LOC 3CH)
                        735 ;
0C00 E5                 736 RXI:    PUSH    H           ;SAVE HL
0C01 F5                 737         PUSH    PSW         ;SAVE PSW
0C02 C5                 738         PUSH    B           ;SAVE BC
0C03 D5                 739         PUSH    D           ;SAVE DE
0C04 3E62               740         MVI     A,DRDMA     ;DISABLE RX DMA
0C06 D3A8               741         OUT     MODE57      ;8257 MODE PORT
0C08 3E18               742         MVI     A,18H       ;RESET RST7.5 F/F
0C0A 30                 743         SIM
0C0B 1604               744         MVI     D,04H       ;D IS RESULT COUNTER
0C0D 2A1020             745         LHLD    LDADR       ;GET LOAD POINTER
0C10 E5                 746         PUSH    H           ;SAVE IT
0C11 E5                 747         PUSH    H           ;SAVE IT AGAIN
0C12 45                 748         MOV     B,L         ;SAVE LSB
0C13 2A1320             749         LHLD    CNADR       ;GET CONSOLE POINTER
0C16 04                 750 RXI1:   INR     B           ;BUMP LOAD POINTER LSB
0C17 78                 751         MOV     A,B         ;GET SET TO TEST
0C18 BD                 752         CMP     L           ;LOAD=CONSOLE?
0C19 CA1E0A             753         JZ      BUFFUL      ;YES, BUFFER FULL
0C1C 15                 754         DCR     D           ;DEC COUNTER
0C1D C2160C             755         JNZ     RXI1        ;NOT DONE, TRY AGAIN
0C20 1605               756         MVI     D,05H       ;RESET COUNTER
0C22 E1                 757         POP     H           ;RESTORE LOAD POINTER
0C23 DB90               758 RXI2:   IN      STAT73      ;READ STATUS
0C25 E608               759         ANI     RXINT       ;TEST RX INT BIT
0C27 CA390C             760         JZ      RXI3        ;DONE, GO FINISH UP
0C2A DB90               761         IN      STAT73      ;READ STATUS AGAIN
0C2C E602               762         ANI     RXIRA       ;IS RESULT READY?
0C2E CA230C             763         JZ      RXI2        ;NO, TEST AGAIN
0C31 DB93               764         IN      RXIR73      ;YES, READ RESULT
0C33 77                 765         MOV     M,A         ;STORE IN BUFFER
0C34 2C                 766         INR     L           ;INC BUFFER POINTER
0C35 15                 767         DCR     D           ;DEC COUNTER
0C36 C3230C             768         JMP     RXI2        ;GET MORE RESULTS
0C39 7A                 769 RXI3:   MOV     A,D         ;GET SET TO TEST
0C3A A7                 770         ANA     A           ;ALL RESULTS?
0C3B CA450C             771         JZ      RXI4        ;YES, SO FINISH UP
0C3E 3600               772         MVI     M,00H       ;NO, LOAD 0 TIL DONE
0C40 2C                 773         INR     L           ;BUMP POINTER
0C41 15                 774         DCR     D           ;DEC COUNTER
0C42 C3390C             775         JMP     RXI3        ;GO AGAIN
0C45 221020             776 RXI4:   SHLD    LDADR       ;UPDATE LOAD POINTER
0C48 3A1520             777         LDA     PRMPT       ;GET MODE INDICATOR
0C4B FE2D               778         CPI     '-'         ;NORMAL MODE?
0C4D CA850C             779         JZ      RXI6        ;YES, CLEAN UP BEFORE RETURN
                        780 ;
                        781 ;       POLL MODE SO CHECK CONTROL BYTE
                        782 ;       IF CONTROL IS A POLL, SET UP SPECIAL TX COMMAND BUFFER
                        783 ;       AND RETURN WITH POLL INDICATOR NOT 0
                        784 ;
0C50 E1                 785         POP     H           ;GET PREVIOUS LOAD ADR POINTER
0C51 7E                 786         MOV     A,M         ;GET IC BYTE FROM BUFFER
```

611001-57

```
0C52 E61E       787         ANI     1EH             ;LOOK AT GOOD FRAME BITS
0C54 C2890C     788         JNZ     RXI5            ;IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
0C57 2C         789         INR     L               ;BYPASS R0 AND R1 IN BUFFER
0C58 2C         790         INR     L
0C59 2C         791         INR     L
0C5A 56         792         MOV     D,M             ;GET ADR BYTE AND SAVE IT IN D
0C5B 2C         793         INR     L
0C5C 7E         794         MOV     A,M             ;GET CNTL BYTE FROM BUFFER
0C5D FE93       795         CPI     SNRMP           ;WAS IT SNRM-P?
0C5F CA6C0C     796         JZ      T1              ;YES, GO SET RESPONSE
0C62 FE11       797         CPI     RR0P            ;WAS IT RR(0)-P?
0C64 C2890C     798         JNZ     RXI5            ;YES, GO SET RESPONSE; OTHERWISE RETURN
0C67 1E11       799         MVI     E,RR0F          ;RR(0)-P SO SET RESPONSE TO RR(0)-F
0C69 C36E0C     800         JMP     TXRET           ;GO FINISH LOADING SPECIAL BUFFER
0C6C 1E73       801 T1:     MVI     E,NSAF          ;SNRM-P SO SET RESPONSE TO NSA-F
0C6E 212020     802 TXRET:  LXI     H,CMD8F1        ;SPECIAL BUFFER ADR
0C71 36C8       806         MVI     M,0C8H          ;LOAD TX FRAME COMMAND
0C73 23         808         INX     H               ;INC POINTER
0C74 3600       809         MVI     M,00H           ;L0=0
0C76 23         810         INX     H               ;INC POINTER
0C77 3600       811         MVI     M,00H           ;L1=0
0C79 23         812         INX     H               ;INC POINTER
0C7A 72         813         MOV     M,D             ;LOAD RCVD ADR BYTE
0C7B 23         814         INX     H               ;INC POINTER
0C7C 73         815         MOV     M,E             ;LOAD RESPONSE CNTL BYTE
0C7D 3E01       816         MVI     A,01H           ;SET POLL INDICATOR NOT 0
0C7F 321620     817         STA     POLIN           ;LOAD POLL INDICATOR
0C82 C3890C     818         JMP     RXI5            ;RETURN
                819
0C85 E1         820 RXI6:   POP     H               ;CLEAN UP STACK IF NORMAL MODE
0C86 C3890C     821         JMP     RXI5            ;RETURN
                822
0C89 CD1A0B     823 RXI5:   CALL    RXDMA           ;RESET DMA CHANNEL
0C8C D1         824         POP     D               ;RESTORE REGISTERS
0C8D C1         825         POP     B
0C8E F1         826         POP     PSW
0C8F E1         827         POP     H
0C90 FB         828         EI                      ;ENABLE INTERRUPTS
0C91 C9         829         RET                     ;RETURN
                830 ;
                831 ;
                832 ;MESSAGE TYPER - ASSUMES MESSAGE STARTS AT HL
                833 ;
                834 ;
0C92 C5         835 TYMSG:  PUSH    B               ;SAVE BC
0C93 7E         836 TYMSG2: MOV     A,M             ;GET ASCII CHR
0C94 23         837         INX     H               ;INC POINTER
0C95 FEFF       838         CPI     0FFH            ;STOP?
0C97 CAA10C     839         JZ      TYMSG1          ;YES, GET SET FOR EXIT
0C9A 4F         840         MOV     C,A             ;SET UP FOR DISPLAY
0C9B CDF805     841         CALL    ECHO            ;DISPLAY CHR
0C9E C3930C     842         JMP     TYMSG2          ;GET NEXT CHR
0CA1 C1         843 TYMSG1: POP     B               ;RESTORE BC
0CA2 C9         844         RET                     ;RETURN
                845 ;
                846 ;
                847 ;SIGNON MESSAGE
                848 ;
```

611001-58

```
0CC1 20
0CC2 FF
                    857 ;
                    858 ;TRANSMITTER INTERRUPT MESSAGES
                    859 ;
0CC3 0D             860 TXIMSG: DB   CR,'TX INT - ',0FFH
0CC4 54582049
0CC8 4E54202D
0CCC 20
0CCD FF
                    861 ;
                    862 ;
                    863 ;TRANSMITTER INTERRUPT ROUTINE
                    864 ;
0CCE E5             865 TXI:    PUSH   H            ;SAVE HL
0CCF F5             866         PUSH   PSW          ;SAVE PSW
0CD0 C5             867         PUSH   B            ;SAVE BC
0CD1 D5             868         PUSH   D            ;SAVE DE
0CD2 3E61           869         MVI    A,DTDMA      ;DISABLE TX DMA
0CD4 D3A8           870         OUT    MODE57       ;8257 MODE PORT
0CD6 1604           871         MVI    D,04H        ;SET COUNTER
0CD8 2A1020         872         LHLD   LDADR        ;GET LOAD POINTER
0CDB E5             873         PUSH   H            ;SAVE IT
0CDC 45             874         MOV    B,L          ;SAVE LSB IN B
0CDD 2A1320         875         LHLD   CNADR        ;GET CONSOLE POINTER
0CE0 04             876 TXI1:   INR    B            ;INC POINTER
0CE1 78             877         MOV    A,B          ;GET SET TO TEST
0CE2 BD             878         CMP    L            ;LOAD=CONSOLE?
0CE3 CAE40A         879         JZ     BUFFUL       ;YES, BUFFER FULL
0CE6 15             880         DCR    D            ;NO, TEST NEXT LOCATION
0CE7 C2E00C         881         JNZ    TXI1         ;TRY AGAIN
0CEA E1             882         POP    H            ;RESTORE LOAD POINTER
0CEB DB92           883         IN     TXIR73       ;READ RESULT
0CED 77             884         MOV    M,A          ;STORE IN BUFFER
0CEE 2C             885         INR    L            ;INR POINTER
0CEF 3600           886         MVI    M,00H        ;EXTRA RESULT SPOTS 0
0CF1 2C             887         INR    L
0CF2 3600           888         MVI    M,00H
0CF4 2C             889         INR    L
0CF5 3600           890         MVI    M,00H
0CF7 2C             891         INR    L
```

611001-59

```
0CF8 3600       892         MVI     M,00H
0CFA 2C         893         INR     L
0CFB 221020     894         SHLD    LDADR           ;UPDATE LOAD POINTER
0CFE CD350B     899         CALL    TXDMA           ;RESET DMA CHANNEL
0D01 D1         900         POP     D               ;RESTORE DE
0D02 C1         901         POP     B               ;RESTORE BC
0D03 F1         902         POP     PSW             ;RESTORE PSW
0D04 E1         903         POP     H               ;RESTORE HL
0D05 FB         904         EI                      ;ENABLE INTERRUPTS
0D06 C9         905         RET                     ;RETURN
                906 ;
                907 ;
                952 ;
                953 ;
                954         END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS


USER SYMBOLS
ADWN   A 0922   AFCMD  A 09CE   BUFFUL A 0AE4   CH0ADR A 00A0   CH0TC  A 00A1   CH1ADR A 00A2   CH1TC  A 00A3
CMD51  A 0027   CMDBF1 A 2020   CMDBUF A 2000   CMDOUT A 0AFB   CMDREC A 0857   CMODE  A 0931   CNADR  A 2013
CNT053 A 009C   CNT153 A 009D   CNT253 A 009E   CNTL51 A 0089   CNTLC  A 0003   CNVBN  A 05BB   COBR   A 000C
COMM   A 0AE5   COMM1  A 0AEA   COMM2  A 0AFF   COMM73 A 0090   CPBF   A 0020   CR     A 000D   CRLF   A 05EB
DEM    A 0000   DEMODE A 2027   DISPY  A 0A39   DISPY1 A 0A4E   DISPY2 A 0A4D   DRDMA  A 0062   DTDMA  A 0061
ECHO   A 05F8   ENDCHK A 0A1B   ENDMA  A 0063   GDWN   A 08FF   GETCH  A 061F   GETCMD A 087D   GRCMD  A 09C4
ILLEG  A 08A7   LDADR  A 2010   LF     A 000A   LKBR1  A 2017   LKBR2  A 2018   LOOPIT A 0861   MDCNT0 A 0036
MDCNT2 A 00B6   MDE51  A 00CE   MODE53 A 009B   MODE57 A 00A8   MONTOR A 0008   NMOUT  A 06C7   NSAF   A 0073
PAR1   A 0B08   PAR2   A 0B0D   PARIN  A 0AAD   PARIN1 A 0AE0   PARIN2 A 0ADC   PARIN3 A 0ABC   PARM73 A 0091
POLIN  A 2016   PRMPT  A 2015   R0PT   A 0AA2   R1PT   A 0AA7   RBCMD  A 097B   RDCMD  A 0971   RDWN   A 08AF
RDY    A 0002   RESBUF A 2000   RESL73 A 0091   ROCMD  A 095D   RPCMD  A 09D8   RR0F   A 0011   RR0P   A 0011
RSCMD  A 0967   RST65  A 20CE   RST75  A 20D4   RXBUF  A 8200   RXD51  A 0088   RXDMA  A 0B1A   RXI    A 0C00
RXI1   A 0C16   RXI2   A 0C23   RXI3   A 0C39   RXI4   A 0C45   RXI5   A 0C89   RXI6   A 0C85   RXIMSG A 0CB8
RXINT  A 0008   RXIR73 A 0093   RXIRA  A 0002   RXS1   A 0A69   RXS2   A 0A7F   RXS3   A 0A8D   RXSORC A 0A62
RXTC   A 41FF   SBCMD  A 0985   SDWN   A 08D7   SIGNON A 0CA3   SLCMD  A 098F   SNRMP  A 0093   SOCMD  A 09A6
SPCMD  A 09E2   SRCMD  A 09BA   SSCMD  A 09B0   START  A 0300   STAT51 A 0089   STAT57 A 00A8   STAT73 A 0090
STKSRT A 20C0   SW     A 0943   T1     A 0C6C   TBUFFL A 0A24   TBUFL  A 0A07   TBUFL1 A 0A0D   TDWN   A 090E
TEST73 A 0092   TFCMD  A 09EC   TFCMD1 A 09F6   TFRET  A 0A36   TLCMD  A 0999   TRUE   A 0000   TRUE1  A 0000
TXBUF  A 8000   TXD51  A 0088   TXDMA  A 0B25   TXDMA1 A 0B42   TXI    A 0CCE   TXI1   A 0CE0   TXIMSG A 0CC3
TXINT  A 0004   TXIR73 A 0092   TXIRA  A 0001   TXPOL  A 094C   TXRET  A 0C6E   TXSORC A 0A47   TXTC   A 81FF
TYMSG  A 0C92   TYMSG1 A 0CA1   TYMSG2 A 0C93   VALDG  A 075E

ASSEMBLY COMPLETE,  NO ERRORS
```

611001-60

2

October 1990

Asynchronous Communication with the 8274 Multiple-Protocol Serial Controller

Order Number: 210311-002

ASYNCHRONOUS COMMUNICATION WITH THE 8274 MULTIPLE-PROTOCOL SERIAL CONTROLLER

CONTENTS

PAGE

INTRODUCTION 2-350

SERIAL-ASYNCHRONOUS DATA LINKS 2-350

MPSC SYSTEM INTERFACE 2-353

PROGRAMMING 2-356

DATA LINK INTERFACE 2-360

APPENDIX A:

Command/Status Details for Asynchronous Communications 2-361

APPENDIX B:

MPSC Polled Transmit/Receive Character Routines 2-367

APPENDIX C:

Interrupt-Driven Transmit/Receive Software 2-370

APPENDIX D:

Application Example Using SDK-86 2-373

REFERENCES 2-386

INTRODUCTION

The 8274 Multiprotocol serial controller (MPSC) is a sophisticated dual-channel communications controller that interfaces microprocessor systems to high-speed serial data links (at speeds to 880K bits per second) using synchronous or asynchronous protocols. The 8274 interfaces easily to most common microprocessors (e.g., 8048, 8051, 8085, 8086, and 8088), to DMA controllers such as the 8237 and 8257, and to the 8089 I/O processor. Both MPSC communication channels are completely independent and can operate in a full-duplex communication mode (simultaneous data transmission and reception).

Communication Functions

The 8274 performs many communications-oriented functions, including:

— Converting data bytes from a microprocessor system into a serial bit stream for transmission over the data link to a receiving system.

— Receiving serial bit streams and reconverting the data into parallel data bytes that can easily be processed by the microprocessor system.

— Performing error checking during data transfers. Error checking functions include computing/transmitting error codes (such as parity bits or CRC bytes) and using these codes to check the validity of received data.

— Operating independently of the system processor in a manner designed to reduce the system overhead involved in data transfers.

System Interface

The MPSC system interface is extremely flexible, supporting the following data transfer modes:

1. Polled Mode. The system processor periodically reads (polls) an 8274 status register to determine when a character has been received, when a character is needed for transmission, and when transmission errors are detected.
2. Interrupt Mode. The MPSC interrupts the system processor when a character has been received, when a character is needed for transmission, and when transmission errors are detected.
3. DMA Mode. The MPSC automatically requests data transfers from system memory for both transmit and receive functions by means of two DMA request signals per serial channel. These DMA request signals may be directly interfaced to an 8237 or 8257 DMA controller or to an 8089 I/O processor.
4. WAIT Mode. The MPSC ready signal is used to synchronize processor data transfers by forcing the processor to enter wait states until the 8274 is ready for another data byte. This feature enables the 8274 to interface directly to an 8086 or 8088 processor by means of string I/O instructions for very high-speed data links.

Scope

This application note describes the use of the 8274 in asynchronous communication modes. Asynchronous communication is typically used to transfer data to/from video display terminals, modems, printers, and other low-to-medium-speed peripheral devices. Use of the 8274 in both interrupt-driven and polled system environments is described. Use of the DMA and WAIT modes are not described since these modes are employed mainly in synchronous communication systems where extremely high data rates are common. Programming examples are written in PL/M-86 (Appendix B and Appendix C). PL/M-86 is executed by the iAPX-86 and iAPX-88 processor families. In addition, PL/M-86 is very similar to PL/M-80 (executed by the MCS-80 and MCS-85 processor families). In addition, Appendix D describes a simple application example using an SDK-86 in an iAPX-86/88 environment.

SERIAL-ASYNCHRONOUS DATA LINKS

A serial asynchronous interface is a method of data transmission in which the receiving and transmitting systems need not be synchronized. Instead of transmitting clocking information with the data, locally generated clocks (16, 32 or 64 times as fast as the data transmission rate) are used by the transmitting and receiving systems. When a character of information is sent by the transmitting system, the character data is framed (preceded and followed) by special START and STOP bits. This framing information permits the receiving system to temporarily synchronize with the data transmission. (Refer to Figure 1 during the following discussion of asynchronous data transmission.)

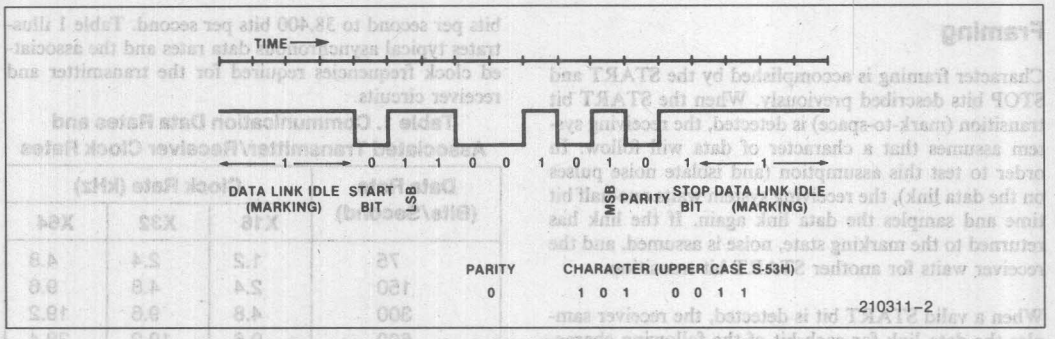


Figure 1. Transmission of a 7-Bit ASCII Character with Even Parity

Normally the data link is in an idle or marking state, continuously transmitting a "mark" (binary 1). When a character is to be sent, the character data bits are immediately preceded by a "space" (binary 0 START bit). The mark-to-space transition informs the receiving system that a character of information will immediately follow the start bit. Figure 1 illustrates the transmission of a 7-bit ASCII character (upper case S) with even parity. Note that the character is transmitted immediately following the start bit. Data bits within the character are transmitted from least-significant to most-significant. The parity bit is transmitted immediately following the character data bits and the STOP framing bit (binary 1) signifies the end of the character.

Asynchronous interfaces are often used with human interface devices such as CRT/keyboard units where the time between data transmissions is extremely variable.

Characters

In asynchronous mode, characters may vary in length from five to eight bits. The character length depends on the coding method used. For example, five-bit characters are used when transmitting Baudot Code, seven-bit characters are required for ASCII data, and eight-bit characters are needed for EBCDIC and binary data. To transmit messages composed of multiple characters, each character is framed and transmitted separately (Figure 2).

This framing method ensures that the receiving system can easily synchronize with the start and stop bits of each character, preventing receiver synchronization errors. In addition, this synchronization method makes both transmitting and receiving systems insensitive to possible time delays between character transmissions.

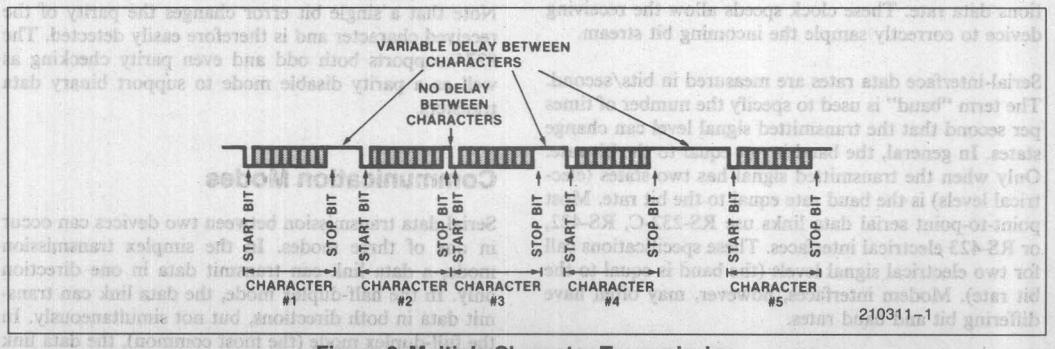


Figure 2. Multiple Character Transmission

Framing

Character framing is accomplished by the START and STOP bits described previously. When the START bit transition (mark-to-space) is detected, the receiving system assumes that a character of data will follow. In order to test this assumption (and isolate noise pulses on the data link), the receiving system waits one-half bit time and samples the data link again. If the link has returned to the marking state, noise is assumed, and the receiver waits for another START bit transition.

When a valid START bit is detected, the receiver samples the data link for each bit of the following character. Character data bits and the parity bit (if required) are sampled at their nominal centers until all required characters are received. Immediately following the data bits, the receiver samples the data link for the STOP bit, indicating the end of the character. Most systems permit specification of 1, 1½, or 2 stop bits.

Timing

The transmitter and receiver in an asynchronous data link arrangement are clocked independently. Normally, each clock is generated locally and the clocks are not synchronized. In fact, each clock may be a slightly different frequency. (In practice, the frequency difference should not exceed a few percent. If the transmitter and receiver clock rates vary substantially, errors will occur because data bits may be incorrectly identified as START or STOP framing bits.) These clocks are designed to operate at 16, 32, or 64 times the communications data rate. These clock speeds allow the receiving device to correctly sample the incoming bit stream.

Serial-interface data rates are measured in bits/second. The term "baud" is used to specify the number of times per second that the transmitted signal level can change states. In general, the baud is not equal to the bit rate. Only when the transmitted signal has two states (electrical levels) is the baud rate equal to the bit rate. Most point-to-point serial data links use RS-232-C, RS-422, or RS-423 electrical interfaces. These specifications call for two electrical signal levels (the baud is equal to the bit rate). Modem interfaces, however, may often have differing bit and baud rates.

While there are generally no limitations on the data transmission rates used in an aysnchronous data link, a limited set of rates has been standardized to promote equipment interconnection. These rates vary from 75 bits per second to 38,400 bits per second. Table 1 illustrates typical asynchronous data rates and the associated clock frequencies required for the transmitter and receiver circuits.

Table 1. Communication Data Rates and Associated Transmitter/Receiver Clock Rates

Data Rate (Bits/Second)	Clock Rate (kHz)		
	X16	X32	X64
75	1.2	2.4	4.8
150	2.4	4.8	9.6
300	4.8	9.6	19.2
600	9.6	19.2	38.4
1200	19.2	38.4	76.8
2400	38.4	76.8	153.6
4800	76.8	153.6	307.2
9600	153.6	307.2	614.2
19200	307.2	614.4	—
38400	614.4	—	—

Parity

In order to detect transmission errors, a parity bit may be added to the character data as it is transferred over the data link. The parity bit is set or cleared to make the total number of "one" bits in the character even (even parity) or odd (odd parity). For example, the letter "A" is represented by the seven-bit ASCII code 1000001 (41H). The transmitted data code (with parity) for this character contains eight bits; 01000001 (41H) for even parity and 11000001 (OC1H) for odd parity. Note that a single bit error changes the parity of the received character and is therefore easily detected. The 8274 supports both odd and even parity checking as well as a parity disable mode to support binary data transfers.

Communication Modes

Serial data transmission between two devices can occur in one of three modes. In the simplex transmission mode, a data link can transmit data in one direction only. In the half-duplex mode, the data link can transmit data in both directions, but not simultaneously. In the full-duplex mode (the most common), the data link can transmit data in both directions simultaneously. The 8274 directly supports the full-duplex mode and will interface to simplex and half-duplex communication data links with appropriate software controls.

BREAK Condition

Asynchronous data links often include a special sequence known as a break condition. A break condition is initiated when the transmitting device forces the data link to a spacing state (binary 0) for an extended length of time (typically 150 milliseconds). Many terminals contain keys to initiate a break sequence. Under software control, the 8274 can initiate a break sequence when transmitting data and detect a break sequence when receiving data.

MPSC SYSTEM INTERFACE

Hardware Environment

The 8274 MPSC interfaces to the system processor over an 8-bit data bus. Each serial I/O channel responds to two I/O or memory addresses as shown in Table 2. In addition, the MPSC supports non-vectored and vectored interrupts.

The 8274 may be configured for memory-mapped or I/O-mapped operation.

The 8274-processor hardware interface can be configured in a flexible manner, depending on the operating mode selected—polled, interrupt-driven, DMA, or WAIT. Figure 3 illustrates typical MPSC configurations for use with an 8088 microprocessor in the polled and interrupt-driven modes.

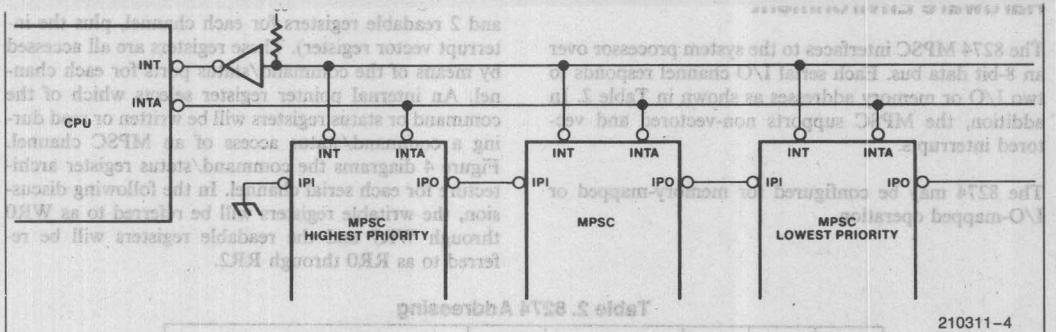
All serial-to-parallel conversion, parallel-to-serial conversion, and parity checking required during asynchronous serial I/O operation is automatically performed by the MPSC.

Operational Interface

Command, parameter, and status information is stored in 21 registers within the MPSC (8 writable registers and 2 readable registers for each channel, plus the interrupt vector register). These registers are all accessed by means of the command/status ports for each channel. An internal pointer register selects which of the command or status registers will be written or read during a command/status access of an MPSC channel. Figure 4 diagrams the command/status register architecture for each serial channel. In the following discussion, the writable registers will be referred to as WR0 through WR7 and the readable registers will be referred to as RR0 through RR2.

Table 2. 8274 Addressing

$\overline{CS}$	A_1	A_0	Read Operation	Write Operation
0	0	0	Ch. A Data Read	Ch. A Data Write
0	1	0	Ch. A Status Read	Ch. A Command/Parameter
0	0	1	Ch. B Data Read	Ch. B Data Write
0	1	1	Ch. B Status Read	Ch. B Command/Parameter
1	X	X	High Impedance	High Impedance



b) Daisy-Chained Interrupt Configuration

Figure 3. 8274 Hardware Interface for Polled and Interrupt-Driven Environments

The least-significant three bits of WR0 are automatically loaded into the pointer register every time WR0 is written. After reset, WR0 is set to zero so that the first write to a command register causes the data to be loaded into WR0 (thereby setting the pointer register). After WR0 is written, the following read or write accesses the register selected by the pointer. The pointer is reset after the read or write operation is completed. In this manner, reading or writing an arbitrary MPSC channel register requires two I/O accesses. The first access is always a write command. This write command is used to set the pointer register. The second access is either a read or a write command; the pointer register (previously set) will ensure that the correct internal register is read or written. After this second access, the pointer register is automatically reset. Note that writing WR0 and reading RR0 does not require presetting of the pointer register.

During initialization and normal MPSC operation, various registers are read and/or written by the system processor. These actions are discussed in detail in the following paragraphs. Note that WR6 and WR7 are not used in the asynchronous communication modes.

RESET

When the 8274 $\overline{\text{RESET}}$ line is activated, both MPSC channels enter the idle state. The serial output lines are forced to the marking state (high) and the modem interface signals ($\overline{\text{RTS}}$, $\overline{\text{DTR}}$) are forced high. In addition, the pointer register is set to zero.

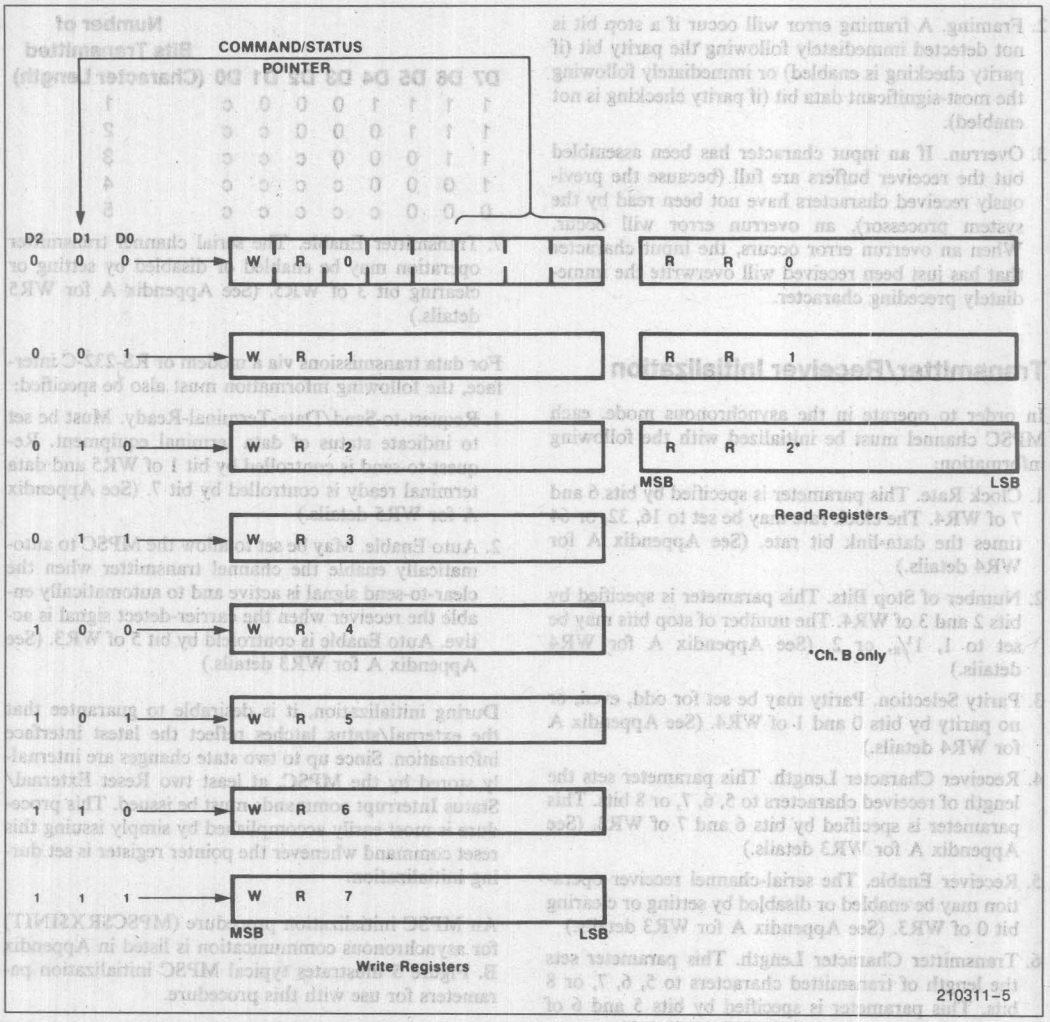


Figure 4. Command/Status Register Architecture (Each Serial Channel)

External/Status Latches

The MPSC continuously monitors the state of four external/status conditions:

1. CTS—clear-to-send input pin.
2. CD—carrier-detect input pin.
3. SYNDET—sync-detect input pin. This pin may be used as a general-purpose input in the asynchronous communication mode.
4. BREAK—a break condition (series of space bits on the receiver input pin).

A change of state in any of these monitored conditions will cause the associated status bit in RR0 (Appendix A) to be latched (and optionally cause an interrupt).

Error Reporting

Three error conditions may be encountered during data reception in the asynchronous mode:

1. Parity. If parity bits are computed and transmitted with each character and the MPSC is set to check parity (bit 0 in WR4 is set), a parity error will occur whenever the number of "1" bits within the character (including the parity bit) does not match the odd/even setting of the parity check flag (bit 1 in WR4).

2. Framing. A framing error will occur if a stop bit is not detected immediately following the parity bit (if parity checking is enabled) or immediately following the most-significant data bit (if parity checking is not enabled).
3. Overrun. If an input character has been assembled but the receiver buffers are full (because the previously received characters have not been read by the system processor), an overrun error will occur. When an overrun error occurs, the input character that has just been received will overwrite the immediately preceding character.

Transmitter/Receiver Initialization

In order to operate in the asynchronous mode, each MPSC channel must be initialized with the following information:

1. Clock Rate. This parameter is specified by bits 6 and 7 of WR4. The clock rate may be set to 16, 32, or 64 times the data-link bit rate. (See Appendix A for WR4 details.)
2. Number of Stop Bits. This parameter is specified by bits 2 and 3 of WR4. The number of stop bits may be set to 1, 1½, or 2. (See Appendix A for WR4 details.)
3. Parity Selection. Parity may be set for odd, even, or no parity by bits 0 and 1 of WR4. (See Appendix A for WR4 details.)
4. Receiver Character Length. This parameter sets the length of received characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 6 and 7 of WR3. (See Appendix A for WR3 details.)
5. Receiver Enable. The serial-channel receiver operation may be enabled or disabled by setting or clearing bit 0 of WR3. (See Appendix A for WR3 details.)
6. Transmitter Character Length. This parameter sets the length of transmitted characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 5 and 6 of WR5. (See Appendix A for WR5 details.) Characters of less than 5 bits in length may be transmitted by setting the transmitted length to five bits (set bits 5 and 6 of WR5 to 1).

   The MPSC then determines the actual number of bits to be transmitted from the character data byte. The bits to be transmitted must be right justified in the data byte, the next three bits must be set to 0 and all remaining bits must be set to 1. The following table illustrates the data formats for transmission of 1 to 5 bits of data:

D7	D6	D5	D4	D3	D2	D1	D0	Number of Bits Transmitted (Character Length)
1	1	1	1	0	0	0	c	1
1	1	1	0	0	0	c	c	2
1	1	0	0	0	c	c	c	3
1	0	0	0	c	c	c	c	4
0	0	0	c	c	c	c	c	5

7. Transmitter Enable. The serial channel transmitter operation may be enabled or disabled by setting or clearing bit 3 of WR5. (See Appendix A for WR5 details.)

For data transmissions via a modem or RS-232-C interface, the following information must also be specified:

1. Request-to-Send/Data-Terminal-Ready. Must be set to indicate status of data terminal equipment. Request-to-send is controlled by bit 1 of WR5 and data terminal ready is controlled by bit 7. (See Appendix A for WR5 details.)
2. Auto Enable. May be set to allow the MPSC to automatically enable the channel transmitter when the clear-to-send signal is active and to automatically enable the receiver when the carrier-detect signal is active. Auto Enable is controleld by bit 5 of WR3. (See Appendix A for WR3 details.)

During initialization, it is desirable to guarantee that the external/status latches reflect the latest interface information. Since up to two state changes are internally stored by the MPSC, at least two Reset External/Status Interrupt commands must be issued. This procedure is most easily accomplished by simply issuing this reset command whenever the pointer register is set during initialization.

An MPSC initialization procedure (MPSCRXINIT) for asynchronous communication is listed in Appendix B. Figure 5 illustrates typical MPSC initialization parameters for use with this procedure.

```
call MPSC$RX$INIT(41,  1,1,0,1,  3,1,1,  3,1,1,0,1);

initializes the 8274 at address 41 as follows:

X16 clock rate              Enable transmitter
1 stop bit                    and receiver
Odd parity                  Auto enable set
8-bit characters            DTR and RTS set
  (Tx and Rx)               Break transmission disabled
```

Figure 5. Sample 8274 Initialization Procedure for Polled Operation

Polled Operation

In the polled mode, the processor must monitor the MPSC status by testing the appropriate bits in the read register. Data available, status, and error conditions are represented in RR0 and RR1 for channels A and B. An example of MPSC-polled transmitter/receiver routines are given in Appendix B. The following routines are detailed:

1. MPSC$POLL$RCV$CHARACTER—This procedure receives a character from the serial data link. The routine waits until the character-available flag in RR0 has been set. When this flag indicates that a character is available, RR1 is checked for errors (overrun, parity, or framing). If an error is detected, the character in the MPSC receive buffer must be read and discarded and the error routine (RECEIVE$ERROR) is called. If no receive errors have been detected, the character is input from the 8274 data port and returned to the calling program.

   MPSC$POLL$RCV$CHARACTER requires three parameters—the address of the 8274 channel data port (data$port), the address of the 8274 channel command port (cmd$port), and the address of a byte variable in which to store the received character (character$ptr).

2. MPSC$POLL$TRAN$CHARACTER—This procedure transmits a character to the serial data link. The routine waits until the transmitter-buffer-empty flag has been set in RR0 before writing the character to the 8274.

   MPSC$POLL$TRAN$CHARACTER requires three parameters—the address of the 8274 channel data port (data$port), the address of the 8274 channel command port (cmd$port), and the character of data that is to be transmitted (character).

3. RECEIVE$ERROR—This procedure processes receiver errors. First, an Error Reset command is written to the affected channel. All additional error processing is dependent on the specific application. For example, the receiving device may immediately request retransmission of the character or wait until a message has been completed.

   RECEIVE$ERROR requires two parameters—the address of the affected 8274 command port (cmd$port) and the error status (status) from 8274 register RR1.

Interrupt-Driven Operation

In an interrupt-driven environment, all receiver operations are reported to the system processor by means of interrupts. Once a character has been received and assembled, the MPSC interrupts the system processor. The system processor must then read the character from the MPSC data buffer and clear the current interrupt. During transmission, the system processor starts serial I/O by writing the first character of a message to the MPSC. The MPSC interrupts the system processor whenever the next character is required (i.e., when the transmitter buffer is empty) and the processor responds by writing the next character of the message to the MPSC data port for the appropriate channel.

By using interrupt-driven I/O, the MPSC proceeds independently of the system processor, signalling the processor only when characters are required for transmission, when characters are received from the data link, or when errors occur. In this manner, the system processor may continue execution of other tasks while serial I/O is performed concurrently.

Interrupt Configurations

The 8274 is designed to interface to 8085- and 8086-type processors in much the same manner as the 8259A is designed. When operating in the 8085 mode, the 8274 causes a "call" to a prespecified, interrupt-service routine location. In the 8086 mode, the 8274 presents the processor with a one-byte interrupt-type number. This interrupt-type number is used to "vector" through the 8086 interrupt service table. In either case, the interrupt service address or interrupt-type number is specified during MPSC initialization.

To shorten interrupt latency, the 8274 can be programmed to modify the prespecified interrupt vector so that no software overhead is required to determine the cause of an interrupt. When this "status affects vector" mode is enabled, the following eight interrupts are differentiated automatically by the 8274 hardware:

1. Channel B Transmitter Buffer Empty.
2. Channel B External/Status Transition.
3. Channel B Character Available.
4. Channel B Receive Error.
5. Channel A Transmitter Buffer Empty.
6. Channel A External/Status Transition.
7. Channel A Character Available.
8. Channel A Receive Error.

Interrupt Sources/Priorities

The 8274 has three interrupt sources for each channel:

1. Receiver (RxA, RxB). An interrupt is initiated when a character is available in the receiver buffer or when a receiver error (parity, framing, or overrun) is detected.
2. Transmitter (TxA, TxB). An interrupt is initiated when the transmitter buffer is empty and the 8274 is ready to accept another character for transmission.

3. External/Status (ExTA, ExTB). An interrupt is initiated when one of the external/status conditions (CDE, CTS, SYNDET, BREAK) changes state.

The 8274 supports two interrupt priority orderings (selectable during MPSC initialization) as detailed in Appendix A, WR2, CH-A.

Interrupt Initialization

In addition to the initialization parameters required for polled operation, the following parameters must be supplied to the 8274 to specify interrupt operation:

1. Transmit Interrupt Enable. Transmitter-buffer-empty interrupts are separately enabled by bit 1 of WR1. (See Appendix A for WR1 details.)
2. Receive Interrupt Enable. Receiver interrupts are separately enabled in one of three modes: a) interrupt on first received character only and on receive errors (used for message-oriented transmission systems), b) interrupt on all received characters and on receive errors, but do not interrupt on parity errors, and c) interrupt on all received characters and on receive errors (including parity errors). The ability to separately disable parity interrupts can be extremely useful when transmitting messages. Since the parity error bit in RR1 is latched, it will not be reset until an error reset operation is performed. Therefore, the parity error bit will be set if any parity errors were detected in a multi-character message. If this mode is used, the serial I/O software must poll the parity error bit at the completion of a message and issue an error reset if appropriate. The receiver interrupt mode is controlled by bits 3 and 4 of WR1. (See Appendix A for WR1 details.)
3. External/Status Interrupts. External/Status interrupts can be separately enabled by bit 0 of WR1. (See Appendix A for WR1 details.)
4. Interrupt Vector. An eight-bit interrupt-service routine location (8085) or interrupt type (8086) is specified through WR2 of channel B. (See Appendix A for WR2 details.) Table 3 lists interrupt vector addresses generated by the 8274 in the "status affects vector" mode.
5. "Status Affects Vector" Mode. The 8274 will automatically modify the interrupt vector if bit 3 of WR1 is set. (See Appendix A for WR1 details.)
6. System Configuration. Specifies the 8274 data transfer mode. Three configuration modes are available: a) interrupt-driven operation for both channels, b) DMA operation for both channels, and c) DMA operation for channel A, interrupt-driven operation for channel B. The system configuration is specified by means of bits 0 and 1 of WR2 (channel A). (See Appendix A for WR2 details.)
7. Interrupt Priorities. The 8274 permits software specification of receive/transmit priorities by means of bit 2 of WR2 (channel A). (See Appendix A for WR2 details.)
8. Interrupt Mode. Specifies whether the MPSC is to operate in a non-vectored mode (for use with an external interrupt controller), in an 8086-vectored mode, or in an 8085-vectored mode. This parameter is specified through bits 3 and 4 of WR2 (channel A). (See Appendix A for WR2 details.)

An MPSC interrupt initialization procedure (MPSCINTINIT) is listed in Appendix C.

Table 3. MPSC-Generated Interrupt Vectors in "Status Affects Vector" Mode

V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0	Original Vector (Specified during Initialization)
8086 Interrupt Type								8085 Interrupt Location								Interrupt Condition
V7	V6	V5	V4	V3	0	0	0	V7	V6	V5	0	0	0	V1	V0	Channel B Transmitter Buffer Empty
V7	V6	V5	V4	V3	0	0	1	V7	V6	V5	0	0	1	V1	V0	Channel B External/Status Change
V7	V6	V5	V4	V3	0	1	0	V7	V6	V5	0	1	0	V1	V0	Channel B Receiver Character Available
V7	V6	V5	V4	V3	0	1	1	V7	V6	V5	0	1	1	V1	V0	Channel B Receive Error
V7	V6	V5	V4	V3	1	0	0	V7	V6	V5	1	0	0	V1	V0	Channel A Transmitter Buffer Empty
V7	V6	V5	V4	V3	1	0	1	V7	V6	V5	1	0	1	V1	V0	Channel A External/Status Change
V7	V6	V5	V4	V3	1	1	0	V7	V6	V5	1	1	0	V1	V0	Channel A Receiver Character Available
V7	V6	V5	V4	V3	1	1	1	V7	V6	V5	1	1	1	V1	V0	Channel A Receive Error

Interrupt Service Routines

Appendix C lists four interrupt service procedures, a buffer transmission procedure, and a buffer reception procedure that illustrate the use of the 8274 in interrupt-driven environments. Use of these procedures assumes that the 8086/8088 interrupt vector is set to 20H and that channel B is used with the "status affects vector" mode enabled.

1. TRANSMIT$BUFFER—This procedure begins serial transmission of a data buffer. Two parameters are required—a pointer to the buffer (buf$ptr) and the length of the buffer (buf$length). The procedure first sets the global buffer pointer, buffer length, and initial index for the transmitter-interrupt service routine and initiates transmission by writing the first character of the buffer to the 8274. The procedure then enters a wait loop until the I/O completion status is set by the transmit-interrupt service routine (MPSC$TRANSMIT$CHARACTER$INT).
2. RECEIVE$BUFFER—This procedure inputs a line (terminated by a line feed) from a serial I/O port. Two parameters are required—a pointer to the input buffer (buf$ptr) and a pointer to the buffer length variable (buf$length$ptr). The buffer length will be set by this procedure when the complete line has been input. The procedure first sets the global buffer pointer and initial index for the receiver interrupt service routine. RECEIVE$BUFFER then enters a wait loop until the I/O completion status is set by the receive interrupt routine (MPSC$RECEIVE-$CHARACTER$INT).
3. MPSC$TRANSMIT$CHARACTER$INT—This procedure is executed when the MPSC Tx-buffer-empty interrupt is acknowledged. If the current transmit buffer index is less than the buffer length, the next character in the buffer is written to the MPSC data port and the buffer pointer is updated. Otherwise, the transmission complete status is posted.
4. MPSC$RECEIVE$CHARACTER$INT—This procedure is executed when a character has been assembled by the MPSC and the MPSC has issued a character-available interrupt. If no input buffer has been set up by RECEIVE$BUFFER, the character is ignored. If a buffer has been set up, but it is full, a receive overrun error is posted. Otherwise, the received character is read from the MPSC data port and the buffer index is updated. Finally, if the received character is a line feed, the reception complete status is posted.
5. RECEIVE$ERROR$INT—This procedure is executed when a receive error is detected. First, the error conditions are read from RR1 and the character currently in the MPSC receive buffer is read and discarded. Next, an Error Reset command is written to the affected channel. All additional error processing is application dependent.
6. EXTERNAL$STATUS$CHANGE$INT—This procedure is executed when an external status condition change is detected. The status conditions are read from RR0 and a Reset External/Status Interrupt command is issued. Further error processing is application dependent.

DATA LINK INTERFACE

Serial Data Interface

Each serial I/O channel within the 8274 MPSC interfaces to two data link lines—one line for transmitting data and one for receiving data. During transmission, characters are converted from parallel data format (as supplied by the system processor or DMA device) into a serial bit stream (with START and STOP bits) and clocked out on the TxD pin. During reception, a serial bit stream is input on the RxD pin, framing bits are stripped out of the data stream, and the resulting character is converted to parallel data format and passed to the system processor or DMA device.

Data Clocking

As discussed previously, the frequency of data transmission/reception on the data link is controlled by the MPSC clock in conjunction with the programmed clock divider (in register WR4). The 8274 is designed to permit all four serial interface lines (TxD and RxD for each channel) to operate at different data rates. Four clock input pins (TxC and RxC for each channel) are available for this function. Note that the clock rate divider spcified in WR4 is used for both RxC and TxC on the appropriate channel; clock rate dividers for each channel are independent.

Modem Control

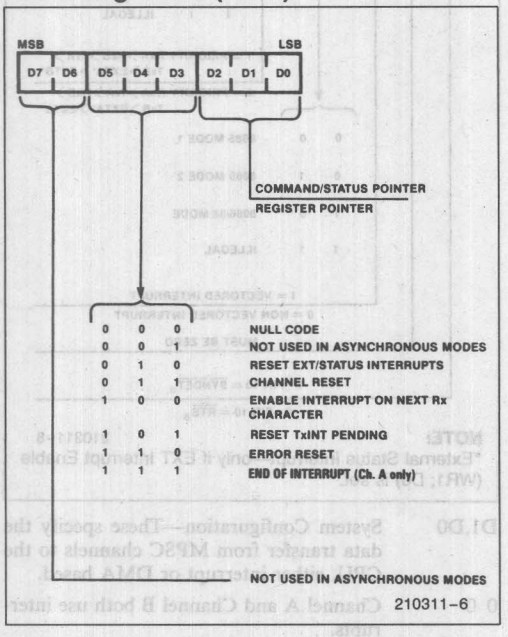
The following four modem interface signals may be connected to the 8274:

1. Data Terminal Ready (DTR). This interface signal (output by the 8274) is software controlled through bit 7 of WR5. When active, DTR indicates that the data terminal/computer equipment is active and ready to interact with the data communications channel. In addition, this signal prepares the modem for connection to the communication channel and maintains connections previously established (e.g., manual call origination).
2. Request To Send (RTS). This interface signal (output by the 8274) is software controlled through bit 1 of WR5. When active, RTS indicates that the data terminal/computer equipment is ready to transmit data. When the RTS bit is reset in asynchronous mode, the signal does not go high until the transmitter is empty.
3. Clear To Send (CTS). This interface signal (input to the 8274) is supplied by the modem in response to an active RTS signal. CTS indicates that the data terminal/computer equipment is permitted to transmit data. The state of CTS is available to the programmer as bit 5 of RR0. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not transmit data bytes until CTS has been activated. If CTS becomes inactive during transmission of a character, the current character transmission is completed before the transmitter is disabled.
4. Carrier Detect (CD). This interface signal (input to the 8274) is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD line. The state of CD is available to the programmer as bit 3 of RR0. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not enable the serial receiver until CD has been activated. If the CD signal becomes inactive during reception of a character, the receiver is disabled, and the partially received character is lost.

In addition to the above modem interface signals, the 8274 SYNDET input pin for channel A may be used as a general-purpose input in the asynchronous communication mode. The status of this signal is available to the programmer as bit 4 of status register RR0.

APPENDIX A
COMMAND/STATUS DETAILS FOR ASYNCHRONOUS COMMUNICATION

Write Register 0 (WR0):



D2,D1,D0 Command/Status Register Pointer bits determine which write-register the next byte is to be written into, or which read-register the next byte is to be read from. After reset, the first byte written into either channel goes into WR0. Following a read or write to any register (except WR0) the pointer will point to WR0.

D5,D4,D3 Command bits determine which of the basic seven commands are to be performed.

Command 0 Null—has no effect.

Command 1 Note used in asynchronous modes.

Command 2 Reset External/Status Interrupts—resets the latched status bits of RR0 and reenables them, allowing interrupts to occur again.

Command 3 Channel Reset—resets the Latched Status bits of RR0, the interrupt prioritization logic and all control registers for the channel. Four extra system clock cycles should be allowed for MPSC reset time before any additional commands or controls are written into the channel.

Command 4 Enable Interrupt on Next Receive Character—if the Interrupt-on-First-Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the MPSC for the next message.

Command 5 Reset Transmitter Interrupt Pending—if the Transmit Interrupt mode is selected, the MPSC automatically interrupts data when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts until the next character has been completely sent.

Command 6 Error Reset—error latches, Parity and Overrun errors in RR1 are reset.

Command 7 End of Interrupt—resets the interrupt-in-service latch of the highest-priority internal device under service.

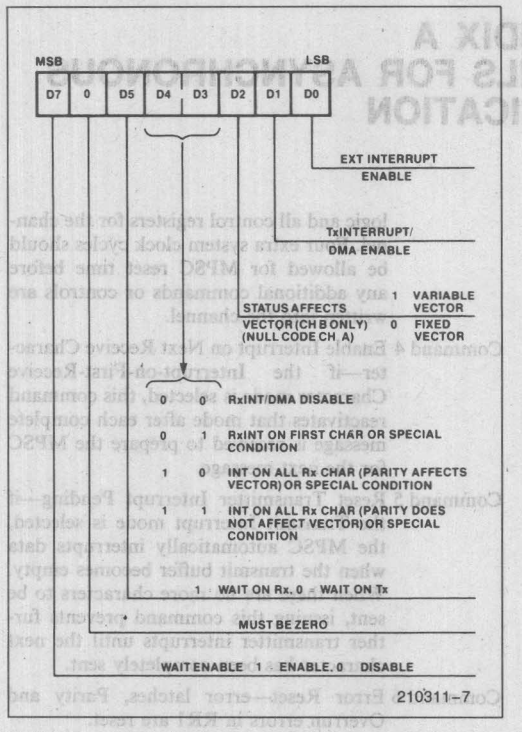
Write Register 1 (WR1):

D0 External/Status Interrupt Enable—allows interrupt to occur as the result of transitions on the $\overline{CD}$, $\overline{CTS}$ or $\overline{SYNDET}$ inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.

D1 Transmitter Interrupt/DMA Enable—allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

D2 Status Affects Vector—(WR1, D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit is set, then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

Write Register 1 (WR1):



D4,D3 Receive Interrupt Mode.

0 0 Receive Interrupts/DMA Disabled.

0 1 Receive Interrupt on First Character Only or Special Condition.

1 0 Interrupt on All Receive Characters of Special Condition (Parity Error is a Special Receive Condition).

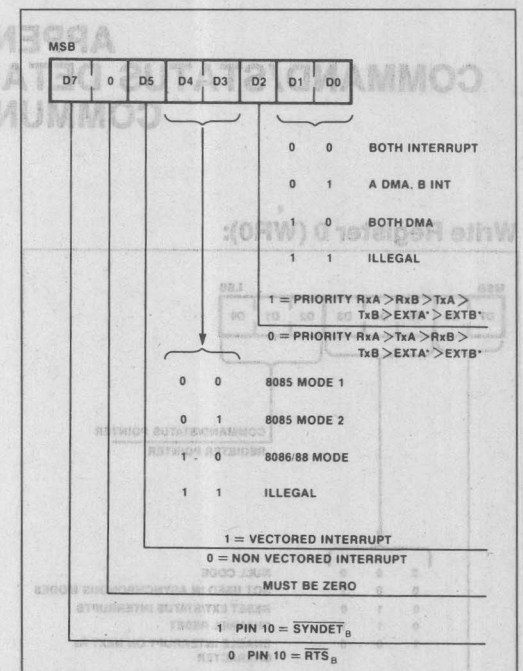
1 1 Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).

D5 Wait on Receive/Transmit—when the following conditions are met, the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, $\overline{CS}$ = 0, A0 = 0/1, and A1 = 0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDY_A and RDY_B may be wired *or* connected since only one signal is active at any one time while the other is in the High Z state.

D6 Must be Zero.

D7 Wait Enable—enables the wait function.

Write Register 2 (WR2): Channel A



NOTE:
*External Status Interrupt—only if EXT Interrupt Enable (WR1; D0) is set.

D1,D0 System Configuration—These specify the data transfer from MPSC channels to the CPU, either interrupt or DMA based.

0 0 Channel A and Channel B both use interrupts.

0 1 Channel A uses DMA, Channel B uses interrupt.

1 0 Channel A and Channel B both use DMA.

1 1 Illegal Code.

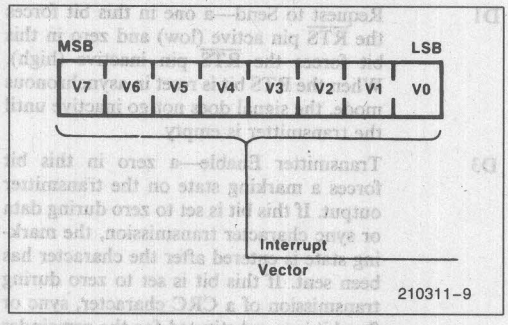
D2 Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.

0 (Highest) RxA, TxA, RxA, RxB, TxBExTA, ExTB (Lowest).

1 (Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest).

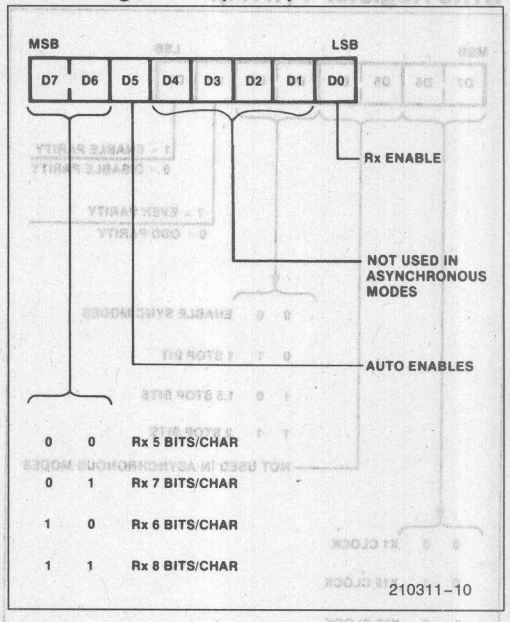
D5,D4,D3	Interrupt Code—specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table.)
0 X X	Non-vectored interrupts—intended for use with an external interrupt controller such as the 8259A.
1 0 0	8085 Vector Mode 1—intended for use as the primary MPSC in a daisy-chained priority structure.
1 0 1	8085 Vector Mode 2—intended for use as any secondary MPSC in a daisy-chained priority structure.
1 1 0	8086/88 Vector Mode—intended for use as either a primary or secondary in a daisy-chained priority structure.
D6	Must be Zero.
D7	
0	Pin 10 = $\overline{RTS}_B$.
1	Pin 10 = $\overline{SYNDET}_B$.

Write Register 2 (WR2): Channel B



D7–D0 Interrupt vector—this register contains the value of the interrupt vector placed on the data bus during acknowledge sequences.

Write Register 3 (WR3):



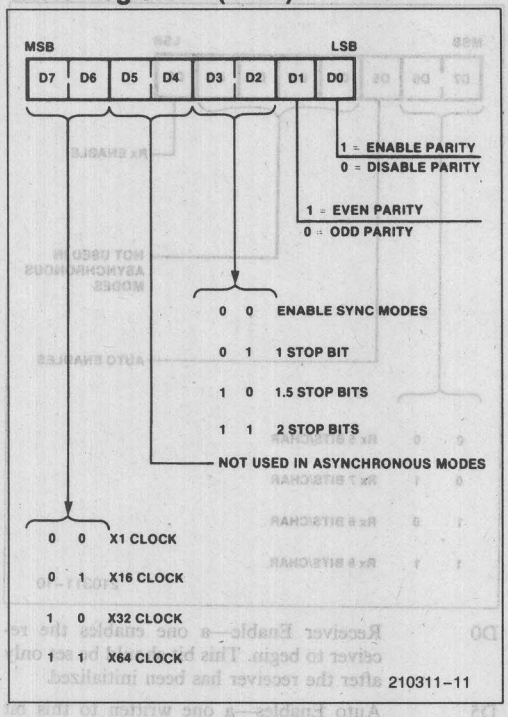
D0 Receiver Enable—a one enables the receiver to begin. This bit should be set only after the receiver has been initialized.

D5 Auto Enables—a one written to this bit causes $\overline{CD}$ to be an automatic enable signal for the receiver and $\overline{CTS}$ to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of $\overline{CD}$ and $\overline{CTS}$ signals to setting/resetting their corresponding bits in the status register (RR0).

D7,D6	Receiver Character length.
0 0	Receive 5 Data bits/character.
0 1	Receive 7 Data bits/character.
1 0	Receive 6 Data bits/character.
1 1	Receive 8 Data bits/character.

2

Write Register 4 (WR4):



D0 Parity—a one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.

D1 Even/Odd Parity—if parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and zero causes it to send and expect odd parity.

D3,D2 Stop Bits.

0 0 Selects synchronous modes.

0 1 Async mode, 1 stop bit/character.

1 0 Async mode, 1½ stop bits/character.

1 1 Async mode, 2 stop bits/character.

D7,D6 Clock mode—selects the clock/data rate multiplier for both the receiver and the transmitter. If the 1x mode is selected, bit synchronization must be done externally.

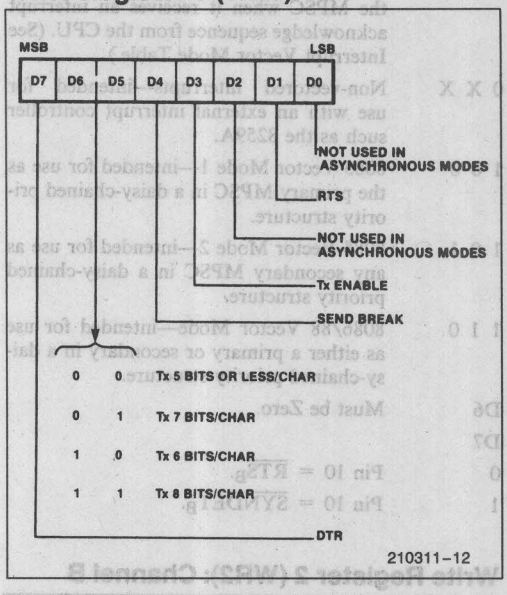
0 0 Clock rate = Data rate × 1.

0 1 Clock rate = Data rate × 16.

1 0 Clock rate = Data rate × 32.

1 1 Clock rate = Data rate × 64.

Write Register 5 (WR5):



D1 Request to Send—a one in this bit forces the $\overline{RTS}$ pin active (low) and zero in this bit forces the $\overline{RTS}$ pin inactive (high). When the RTS bit is reset in asynchronous mode, the signal does not go inactive until the transmitter is empty.

D3 Transmitter Enable—a zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.

D4 Send Break—a one in this bit forces the transmit data low. A zero in this bit allows normal transmitter operation.

D6,D5 Transmit Character length.

0 0 Transmit 5 or less bits/character.

0 1 Transmit 7 bits/character.

1 0 Transmit 6 bits/character.

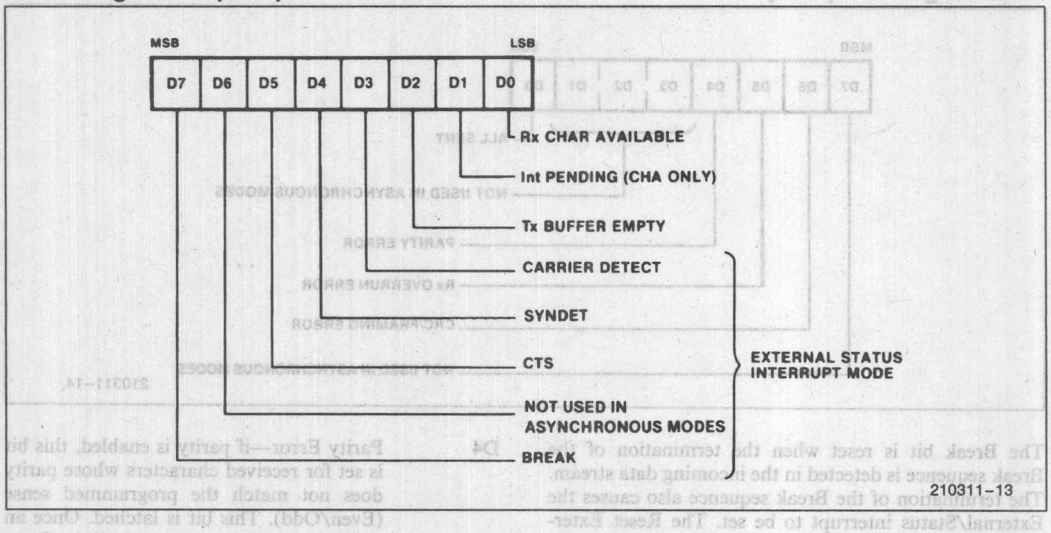
1 1 Transmit 8 bits/character.

Bits to be sent must be right justified, least-significant bit first, e.g.:

D7 D6 D5 D4 D3 D2 D1 D0

0 0 B5 B4 B3 B2 B1 B0

Read Register 0 (RR0):



D0 Receive Character Available—this bit is set when the receive FIFO contains data and is reset when the FIFO is empty.

D1 Interrupt Pending—This Interrupt-Pending bit is reset when an E01 command is issued and there is no other interrupt request pending at that time. In vector mode, this bit is set at the falling edge of the second INTA in an INTA cycle for an internal interrupt request. In non-vector mode, this bit is set at the falling edge of RD input after pointer 2 is specified. This bit is always zero in Channel B.

D2 Transmit Buffer Empty—This bit is set whenever the transmit buffer is empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.

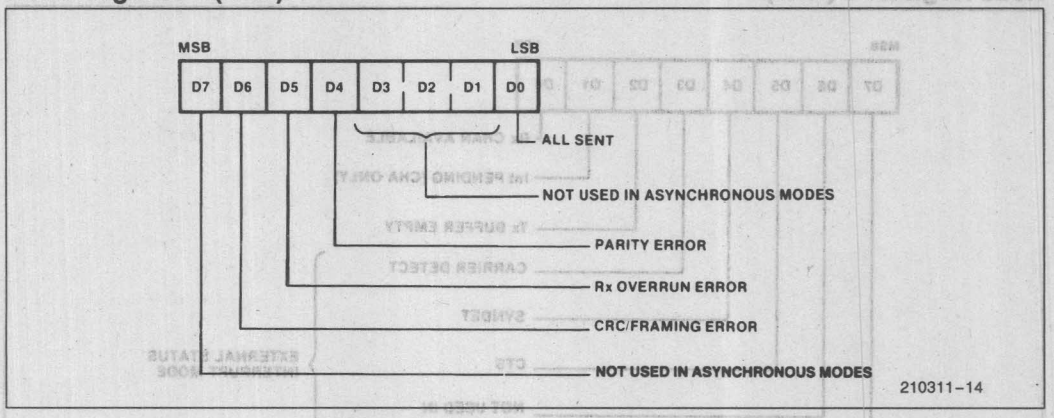
D3 Carrier Detect—This bit contains the state of the $\overline{CD}$ pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the $\overline{CD}$ pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the $\overline{CD}$ pin immediately following a Reset External/Status Interrupt command.

D4 SYNDET—In asynchronous modes, the operation of this bit is similar to the CD status bit, except that it shows the state of the $\overline{SYNDET}$ input. Any High-to-Low transition on the $\overline{SYNDET}$ pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the $\overline{SYNDET}$ pin at time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the $\overline{SYNDET}$ input.

D5 Clear to Send—this bit contains the inverted state of the $\overline{CTS}$ pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the $\overline{CTS}$ pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the $\overline{CTS}$ pin immediately following a Reset External/Status Interrupt command.

D7 Break—in the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WR0, Command 2) to the break detection logic so the Break sequence termination can be recognized.

Read Register 1 (RR1):



The Break bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single, extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

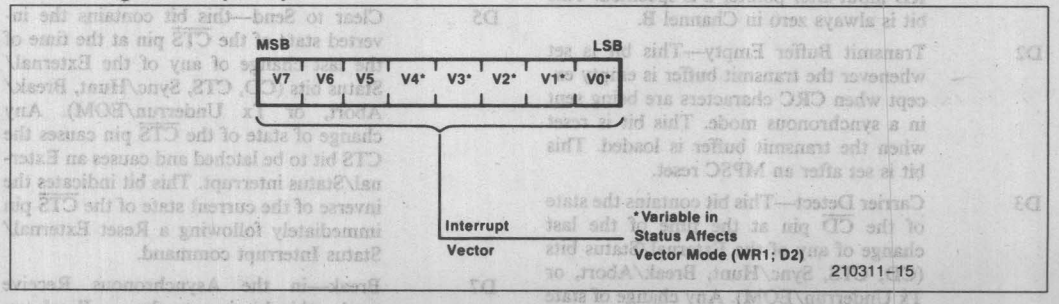
D0 All sent—this bit is set when all characters have been sent. It is reset when characters are in the transmitter. In synchronous modes, this bit is always set.

D4 Parity Error—if parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

D5 Receive Overrun Error—this bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwrittenand flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the "status affects vector" mode, the overrun causes a Special Receive Error Vector.

D6 Framing Error—in async modes, a one in this bit indicates a receive framing error. It can be reset by issuing an Error Reset command.

Read Register 2 (RR2):



D7–D0 Interrupt vector—contains the interrupt vector programmed into WR2. If the "status affects vector" mode is selected, it contains the modified vector. (See WR2.) RR2 contains the modified vector for the highest priority interrupt pending. If no interrupts are pending, the variable bits in the vector are set to one. May be read from Channel B only.

RR2 Channel B

APPENDIX B
MPSC-POLLED TRANSMIT/RECEIVE CHARACTER ROUTINES

```
MPSC$RX$INIT: procedure (cmd$port,
                         clock$rate,stop$bits,parity$type,parity$enable,
                         rx$char$length,rx$enable,auto$enable,
                         tx$char$length,tx$enable,dtr,brk,rts);

  declare cmd$port       byte,
          clock$rate     byte,
          stop$bits      byte,
          parity$type    byte,
          parity$enable  byte,
          rx$char$length byte,
          rx$enable      byte,
          auto$enable    byte,
          tx$char$length byte,
          tx$enable      byte,
          dtr            byte,
          brk            byte,
          rts            byte;


  output(cmd$port)=30H;          /* channel reset */

  output(cmd$port)=14H;          /* point to WR4 */
  /* set clock rate, stop bits, and parity information */
  output(cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)
                   or parity$enable;

  output(cmd$port)=13H;          /* point to WR3 */
  /* set up receiver parameters */
  output(cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);

  output(cmd$port)=15H;          /* point to WR5 */
  /* set up transmitter parameters */
  output(cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)
                   or shl(brk,4) or shl(rts,1);

  end MPSC$RX$INIT;
```

210311-16

```
      return 0;                                  /* error return - no character avail */
      end;
    else do;
      character=input(data$port);
      return 0FFH;                               /* good return - character avail */
      end;

end MPSC$POLL$RCV$CHARACTER;


MPSC$POLL$TRAN$CHARACTER: procedure(data$port,cmd$port,character);

  declare data$port      byte,
          cmd$port       byte,
          character      byte;

  declare tx$buffer$empty literally '4';

  /* wait for transmitter buffer empty */
  while not (input(cmd$port) and tx$buffer$empty) do; end;

  /* output character */
  output(data$port)=character;

end MPSC$POLL$TRAN$CHARACTER;


RECEIVE$ERROR: procedure(cmd$port,status);

  declare cmd$port       byte,
          status         byte;

  output(cmd$port)=30H;                  /* error reset */

  /* *** other application dependent
        error processing should be placed here   *** */

end RECEIVE$ERROR;
```

210311-17

```
TRANSMIT$BUFFER: procedure(buf$ptr,buf$length)

  declare
    buf$ptr       pointer,
    buf$length    byte;

  /* set up transmit buffer pointer and buffer length in global variables for
     interrupt service */
  tx$buffer$ptr=buf$ptr;
  transmit$length=buf$length;

  transmit$status=not$complete;              /* setup status for not complete */
  output(data$port)=transmit$buffer(0);      /* transmit first character */
  transmit$index=1;                          /* first character transmitted */

  /* wait until transmission complete or error detected */
  while transmit$status = not$complete do; end;
  if transmit$status <> complete
    then return false;
    else return true;

end TRANSMIT$BUFFER;


RECEIVE$BUFFER: procedure (buf$ptr,buf$length$ptr);

  declare
    buf$ptr         pointer,
    buf$length$ptr  pointer,
    buf$length      based buf$length$ptr byte;

  /* set up receive buffer pointer in global variable for interrupt service */
  rx$buffer$ptr=buf$ptr;
  receive$index=0;

  receive$status=not$complete;           /* set status to not complete */
  /* wait until buffer received */
  while receive$status = not$complete do; end;
  buf$length=receive$length;
  if receive$status = complete
    then return true;
    else return false;

end RECEIVE$BUFFER;
```

210311-18

2

APPENDIX C
INTERRUPT-DRIVEN TRANSMIT/RECEIVE SOFTWARE

```
declare
  /* global variables for buffer manipulation */

  rx$buffer$ptr          pointer,              /* pointer to receive buffer */
  receive$buffer based rx$buffer$ptr(128) byte,
  receive$status         byte initial(0),      /* indicates receive buffer status */
  receive$index          byte,                 /* current index into receive buffer */
  receive$length         byte,                 /* length of final receive buffer */

  tx$buffer$ptr          pointer,              /* pointer to transmit buffer */
  transmit$buffer based tx$buffer$ptr(128) byte,
  transmit$status        byte initial(0),      /* indicates transmit buffer status */
  transmit$index         byte,                 /* current index into transmit buffer */
  transmit$length        byte,                 /* length of buffer to be transmitted */

  cmd$port               literally '43H',
  data$port              literally '41H',
  a$cmd$port             literally '42H',
  b$cmd$port             literally '43H',
  line$feed              literally '0AH',
  not$complete           literally '0',
  complete               literally '0FFH',
  overrun                literally '1',

  channel$reset          literally '18H',
  error$reset            literally '30H',
  reset$ext$status       literally '10H';
```

210311-20

```
MPSC$INT$INIT: procedure (clock$rate,stop$bits,parity$type,parity$enable,
                          rx$char$length,rx$enable,auto$enable,
                          tx$char$length,tx$enable,dtr,brk,rts,
                          ext$en,tx$en,rx$en,stat$affects$vector,
                          config,priority,vector$int$mode,int$vector);

  declare
        clock$rate        byte,        /* 2-bit code for clock rate divisor */
        stop$bits         byte,        /* 2-bit code for number of stop bits */
        parity$type       byte,        /* 1-bit parity type */
        parity$enable     byte,        /* 1-bit parity enable */
        rx$char$length    byte,        /* 2-bit receive character length */
        rx$enable         byte,        /* 1-bit receiver enable */
        auto$enable       byte,        /* 1-bit auto enable flag */
        tx$char$length    byte,        /* 2-bit transmit character length */
        tx$enable         byte,        /* 1-bit transmitter enable */
        dtr               byte,        /* 1-bit status of DTR pin */
        brk               byte,        /* 1-bit data link break enable */
        rts               byte,        /* 1-bit status of RTS pin */
        ext$en            byte,        /* 1-bit external/status enable */
        tx$en             byte,        /* 1-bit Tx interrupt enable */
        rx$en             byte,        /* 2-bit Rx interrupt enable/mode */
        stat$aff$vector   byte,        /* 1-bit status affects vector flag */
        config            byte,        /* 2-bit system config - int/DMA */
        priority          byte,        /* 1-bit priority flag */
        vector$int$mode   byte,        /* 3-bit interrupt mode code */
        int$vector        byte;        /* 8-bit interrupt type code */


  output(b$cmd$port)=channel$reset;  /* channel reset */

  output(b$cmd$port)=14H;            /* point to WR4 */
  /* set clock rate, stop bits, and parity information */
  output(b$cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)
                     or parity$enable;

  output(b$cmd$port)=13H;            /* point to WR3 */
  /* set up receiver parameters */
  output(b$cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);

  output(b$cmd$port)=15H;            /* point to WR5 */
  /* set up transmitter parameters */
  output(b$cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)
                   or shl(brk,4) or shl(rts,1);

  output(b$cmd$port)=12H;            /* point to WR2 */
  /* set up interrupt vector */
  output(b$cmd$port)=int$vector;

  output(a$cmd$port)=12H;            /* point to WR2, channel A */
  /* set up interrupt modes */
  output(a$cmd$port)=shl(vector$int$mode,3) or shl(priority,2) or config;

  output(b$cmd$port)=11H;            /* point to WR1 */
  /* set up interrupt enables */
  output(b$cmd$port)=shl(rx$en,3) or shl(stat$aff$vector,2) or shl(tx$en,1)
                     or ext$en;

  end MPSC$INT$INIT;
```

210311-21

```
MPSC$RECEIVE$CHARACTER$INT: procedure interrupt 22H;

  /* ignore input if no open buffer */
  if receive$status <> not$complete then return;

  /* check for receive buffer overrun */
  if receive$index = 128
    then receive$status=overrun;
    else do;
      /* read character from MPSC and place in buffer - note that the
         parity of the character must be masked off during this step if
         the character is less than 8 bits (e.g., ASCII)  */
      receive$buffer(receive$index),character=input(data$port) and 7FH;
      receive$index=receive$index+1;          /* update receive buffer index */

      /* check for line feed to end line */
      if character = line$feed
        then do; receive$length=receive$index; receive$status=complete; end;
    end;

end MPSC$RECEIVE$CHARACTER$INT;


MPSC$TRANSMIT$CHARACTER$INT: procedure interrupt 20H;

  /* check for more characters to transfer */
  if transmit$index < transmit$length
    then do;
      /* write next character from buffer to MPSC */
      output(data$port)=transmit$buffer(transmit$index);
      transmit$index=transmit$index+1;      /* update transmit buffer index */
    end;
    else transmit$status=complete;

end MPSC$TRANSMIT$CHARACTER$INT;


RECEIVE$ERROR$INT: procedure interrupt 23H;

  declare
    temp                  byte;    /* temporary character storage */

  output(cmd$port)=1;              /* point to RR1 */
  receive$status=input(cmd$port);
  temp=input(data$port);           /* discard character */
  output(cmd$port)=error$reset;    /* send error reset */

  /* *** other application dependent
         error processing should be placed here   *** */

end RECEIVE$ERROR$INT;


EXTERNAL$STATUS$CHANGE$INT: procedure interrupt 21H;

  transmit$status=input(cmd$port)      /* input status change information */
  output(cmd$port)=reset$ext$status;

  /* *** other application dependent
         error processing should be placed here   *** */

end EXTERNAL$STATUS$CHANGE$INT;
```

210311-19

APPENDIX D
APPLICATION EXAMPLE USING SDK-86

This application example shows the 8274 in a simple iAPX-86/88 system. The 8274 controls two separate asynchronous channels using its internal interrupt controller to request all data transfers. The 8274 driver software is described which transmits and receives data buffers provided by the CPU. Also, status registers are maintained in system memory to allow the CPU to monitor progress of the buffers and error conditions.

THE HARDWARE INTERFACE

Nothing could be easier than the hardware design of an interrupt-driven 8274 system. Simply connect the data bus lines, a few bus control lines, supply a timing clock for baud rate and, voila, it's done! For this example, the ubiquitous SDK-86 is used as the host CPU system. The 8274 interface is constructed on the wire-wrap area provided. While discussing the hardware interface, please refer to Diagram 1.

Placing the 8274 on the lower 8 bits of the 8086 data bus allows byte-wide data transfers at even I/O addresses. For simplicity, the 8274's $\overline{CS}$ input is generated by combining the M/IO select line with address line A7 via a 7432. This places the 8274 address range in multiple spots within the 8086 I/O address space. (While fine for this example, a more complete address decoding is recommended for actual prototype systems.) The 8086's A1 and A2 address lines are connected to the A0 and A1 8274 register select inputs respectively. Although other port assignments are possible because of the overlapping address spaces, the following I/O port assignments are used in this example:

Port Function	I/O Address
Data channel A	0000H
Command/status A	0002H
Data channel B	0004H
Command/status B	0006H

To connect the 8274's interrupt controller into the system an inverter and pull-up resistor are needed to convert the 8274's active-low, interrupt-request output, $\overline{INT}$, into the correct polarity for the 8086's INTR interrupt input. The 8274 recognizes interrupt-acknowledge bus cycles by connecting the $\overline{INTA}$ (INTerrupt Acknowledge) lines of the 8274 and 8086 together.

The 8274 ReaD and WRite lines directly connect to the respective 8086 lines. The RESET line requires an inverter. The system clock for the 8274 is provided by the PCLK (peripheral clock) output of the 8284A clock generator.

On the 8274's serial side, traditional 1488 and 1489 RS-232 drivers and receivers are used for the serial interface. The onboard baud rate generator supplies the channel baud rate timing. In this example, both sides of both channels operate at the same baud rate although this certainly is not a requirement. (On the SDK-86, the baud rate selection is hard-wired thru jumpers. A more flexible approach would be to incorporate an 8253 Programmable Interval Timer to allow software-configurable baud rate selection.)

That's all there is to it. This hardware interface is completely general-purpose and supports all of the 8274 features except the DMA data transfer mode which requires an external DMA controller. Now let's look at the software interface.

SOFTWARE INTERFACE

In this example, it is assumed that the 8086 has better things to do rather than continuously run a serial channel. Presenting the software as a group of callable procedures lets the designer include them in the main body of another program. The interrupt-driven data transfers give the effect that the serial channels are handled in the background while the main program is executing in the foreground. There are five basic procedures: a serial channel initialization routine and buffer handling routines for the transmit and receive data buffers of each channel. Appendix D-1 shows the entire software listing. Listing line numbers are referenced as each major routing is discussed.

The channel initialization routine (INITIAL 8274), starting with line #203, simply sets each channel into a particular operating mode by loading the command registers of the 8274. In normal operation, once these registers are loaded, they are rarely changed. (Although this example assumes a simple asynchronous operating mode, the concept is easily extended for the byte- and bit-synchronous modes.)

(For detailed description on SDK-86, refer to SDK-86 MCS-86 System Design Kit Assembly Manual

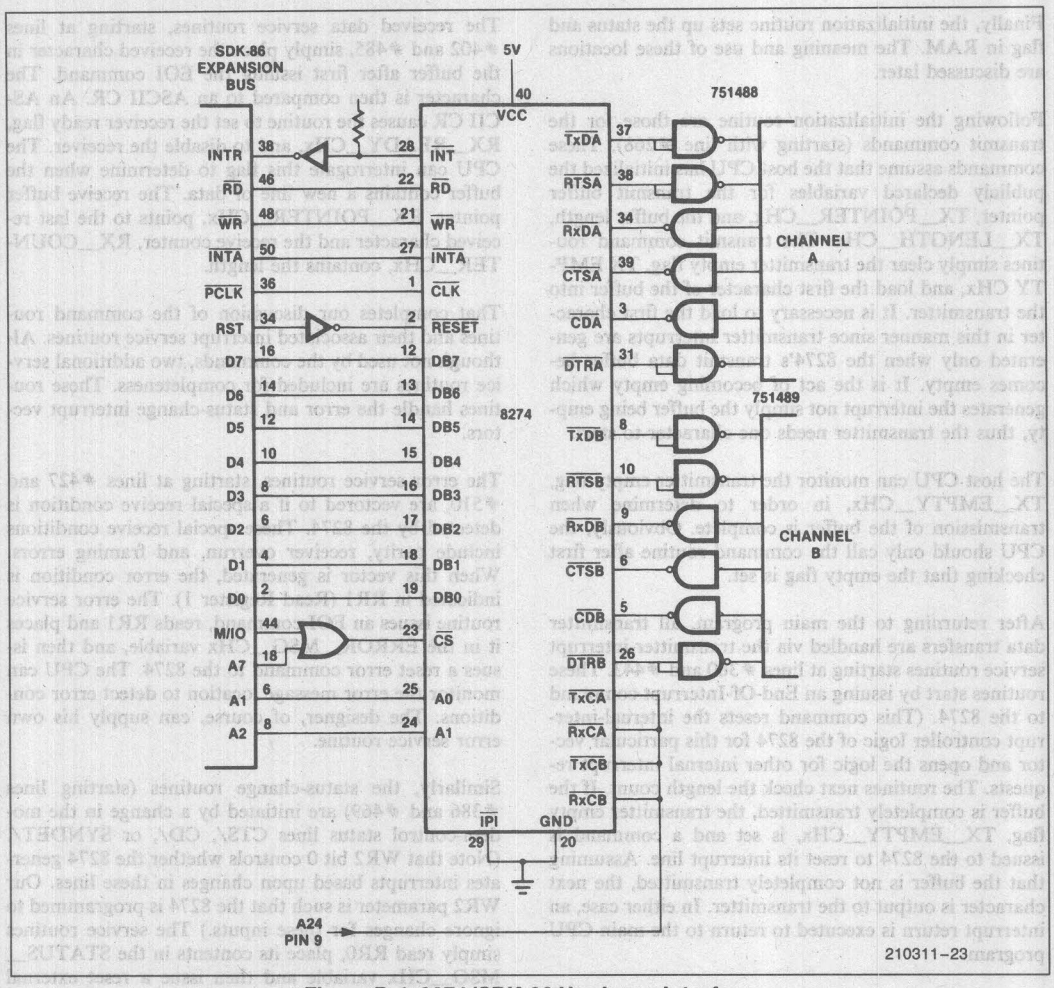


Figure D-1. 8274/SDK-86 Hardware Interface

The channel operating modes are contained in two tables starting with line #163. As the 8274 has only one command register per channel, the remaining seven registers are loaded indirectly through the WR0 (Write Register 0) register. The first byte of each table entry is the register pointer value which is loaded into WR0 and the second byte is the value for that particular register.

The indicated modes set the 8274 for asynchronous operation with data characters 8 bits long, no parity, and 2 stop bits. An X16 baud rate clock is assumed. Also selected is the "interrupt on all RX character" mode with a variable interrupt vector compatible with the 8086/8088. The transmitters are enabled and all model control lines are put in their active state.

In addition to initializing the 8274, this routine also sets up the appropriate interrupt vectors. The 8086 assumes the first 1K bytes of memory contain up to 256 separate interrupt vectors. On the SDK-86 the initial 2K bytes of memory is RAM and therefore must be initialized with the appropriate vectors. (In a prototype system, this initial memory is probably ROM, thus the vector set-up is not needed.) The 8274 supplies up to eight different interrupt vectors. These vectors are developed from internal conditions such as data requests, status changes, or error conditions for each channel. The initialization routine arbitrarily assumes that the initial 8274 vector corresponds to 8086 vector location 80H (memory location 200H). This choice is arbitrary since the 8274 initial vector location is programmable.

Finally, the initialization routine sets up the status and flag in RAM. The meaning and use of these locations are discussed later.

Following the initialization routine are those for the transmit commands (starting with line #268). These commands assume that the host CPU has initialized the publicly declared variables for the transmit buffer pointer, TX__POINTER__CHx, and the buffer length, TX__LENGTH__CHx. The transmit command routines simply clear the transmitter empty flag, TX EMPTY CHx, and load the first character of the buffer into the transmitter. It is necessary to load the first character in this manner since transmitter interrupts are generated only when the 8274's transmit data buffer becomes empty. It is the act of becoming empty which generates the interrupt not simply the buffer being empty, thus the transmitter needs one character to start.

The host CPU can monitor the transmitter empty flag, TX__EMPTY__CHx, in order to determine when transmission of the buffer is complete. Obviously, the CPU should only call the command routine after first checking that the empty flag is set.

After returning to the main program, all transmitter data transfers are handled via the transmitter-interrupt service routines starting at lines #360 and #443. These routines start by issuing an End-Of-Interrupt command to the 8274. (This command resets the internal-interrupt controller logic of the 8274 for this particular vector and opens the logic for other internal interrupt requests. The routines next check the length count. If the buffer is completely transmitted, the transmitter empty flag, TX__EMPTY__CHx, is set and a command is issued to the 8274 to reset its interrupt line. Assuming that the buffer is not completely transmitted, the next character is output to the transmitter. In either case, an interrupt return is executed to return to the main CPU program.

The receiver commands start at line #314. Like the transmit commands, it is assumed that the CPU has initialized the receive-buffer-pointer public variable, RX__POINTER__CHx. This variable points to the first location in an empty receive buffer. The command routines clear the receiver ready flag, RX__READY__CHx, and then set the receiver enable bit in the 8274 WR3 register. With the receiver now enabled, any received characters are placed in the receive buffer using interrupt-driven data transfers.

The received data service routines, starting at lines #402 and #485, simply place the received character in the buffer after first issuing the EOI command. The character is then compared to an ASCII CR. An ASCII CR causes the routine to set the receiver ready flag, RX__READY__CHx, and to disable the receiver. The CPU can interrogate this flag to determine when the buffer contains a new line of data. The receive buffer pointer, RX__POINTER__CHx, points to the last received character and the receive counter, RX__COUNTER__CHx, contains the length.

That completes our discussion of the command routines and their associated interrupt service routines. Although not used by the commands, two additional service routines are included for completeness. These routines handle the error and status-change interrupt vectors.

The error service routines, starting at lines #427 and #510, are vectored to if a special receive condition is detected by the 8274. These special receive conditions include parity, receiver overrun, and framing errors. When this vector is generated, the error condition is indicated in RR1 (Read Register 1). The error service routine issues an EOI command, reads RR1 and places it in the ERROR__MSG__CHx variable, and then issues a reset error command to the 8274. The CPU can monitor the error message location to detect error conditions. The designer, of course, can supply his own error service routine.

Similarly, the status-change routines (starting lines #386 and #469) are initiated by a change in the modem-control status lines CTS/, CD/, or SYNDET/. (Note that WR2 bit 0 controls whether the 8274 generates interrupts based upon changes in these lines. Our WR2 parameter is such that the 8274 is programmed to ignore changes for these inputs.) The service routines simply read RR0, place its contents in the STATUS__MSG__CHx variable and then issue a reset external status command. Read Register 0 contains the state of the modem inputs at the point of the last change.

Well, that's it. This application example has presented useful, albeit very simple, routines showing how the 8274 might be used to transmit and receive buffers using an asynchronous serial format. Extensions for byte- or bit-synchronous formats would require no hardware changes due to the highly programmable nature of the 8274's serial formats.

8274 APPLICATION BRIEF PROGRAM

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE ASYNCB
OBJECT MODULE PLACED IN :F1:ASYNCB.OBJ
ASSEMBLER INVOKED BY:  ASM86 :F1:ASYNCB.SRC


LOC  OBJ                  LINE     SOURCE

                             1     ;*************************************************************
                             2     ;*                                                           *
                             3     ;*                  8274 APPLICATION BRIEF PROGRAM           *
                             4     ;*                                                           *
                             5     ;*                                                           *
                             6     ;*                                                           *
                             7     ;* THE 8274 IS INITIALIZED FOR SIMPLE ASYNCHRONOUS SERIAL    *
                             8     ;* FORMAT AND VECTORED INTERRUPT-DRIVEN DATA TRANSFERS.      *
                             9     ;* THE INITIALIZATION ROUTINE ALSO LOADS THE 8086'S INTERRUPT *
                            10     ;* VECTOR TABLE FROM THE CODE SEGMENT INTO LOW RAM ON THE     *
                            11     ;* SDK-86.  THE TRANSMITTER AND RECEIVER ARE LEFT ENABLED.   *
                            12     ;*                                                           *
                            13     ;* FOR TRANSMIT, THE CPU PASSES IN MEMORY THE POINTER OF A   *
                            14     ;* BUFFER TO TRANSMIT AND THE BYTE LENGTH OF THE BUFFER.     *
                            15     ;* THE DATA TRANSFER PROCEED USING INTERRUPT-DRIVEN TRANSFERS.*
                            16     ;* A STATUS BIT IN MEMORY IS SET WHEN IF BUFFERS IS EMPTY.   *
                            17     ;*                                                           *
                            18     ;* FOR RECEIVE, THE CPU PASSES THE POINTER OF A BUFFER TO FILL.*
                            19     ;* THE BUFFER IS FILLED UNTIL A 'CR_CHR' CHARACTER IS RECEIVED.*
                            20     ;* A STATUS BIT IS SET AND THE CPU MAY READ THE RX POINTER TO *
                            21     ;* DETERMINE THE LOCATION OF THE LAST CHARACTER.             *
                            22     ;*                                                           *
                            23     ;* ALL ROUTINES ARE ASSUMED TO EXIST IN THE SAME CODE SEGMENT.*
                            24     ;* CALL'S TO THE SERVICE ROUTINES ARE ASSUMED TO BE "SHORT" OR *
                            25     ;* INTRASEGMENT (ONLY THE RETURN ADDRESS IP IS ON THE STACK).*
                            26     ;*                                                           *
                            27     ;*                                                           *
                            28     ;*                                                           *
                            29     ;*                                                           *
                            30     ;*************************************************************
```

210311-24

```
                        45          PUBLIC  RX_READY_CHA    ;RX READY FLAG CHA
                        46          PUBLIC  TX_EMPTY_CHB    ;TX EMPTY FLAG CHB
                        47          PUBLIC  TX_EMPTY_CHA    ;TX EMPTY FLAG CHA
                        48          PUBLIC  RX_COUNT_CHB    ;RX BUFFER COUNTER CHB
                        49          PUBLIC  RX_COUNT_CHA    ;RX BUFFER COUNTER CHA
                        50          PUBLIC  ERROR_MSG_CHB   ;ERROR FLAG CHB
                        51          PUBLIC  ERROR_MSG_CHA   ;ERROR FLAG CHA
                        52          PUBLIC  STATUS_MSG_CHB  ;STATUS FLAG CHB
                        53          PUBLIC  STATUS_MSG_CHA  ;STATUS FLAG CHA
                        54
                        55      ;PUBLIC DECLARATIONS FOR VARIABLES PASSED TO THE TRANSMIT
                        56      ;AND RECEIVE COMMANDS
                        57
                        58          PUBLIC  TX_POINTER_CHB  ;TX BUFFER POINTER FOR CHB
                        59          PUBLIC  TX_LENGTH_CHB   ;TX LENGTH OF BUFFER FOR CHB
                        60          PUBLIC  TX_POINTER_CHA  ;TX BUFFER POINTER FOR CHA
                        61          PUBLIC  TX_LENGTH_CHA   ;TX LENGTH OF BUFFER FOR CHA
                        62          PUBLIC  RX_POINTER_CHB  ;RX BUFFER POINTER FOR CHB
                        63          PUBLIC  RX_POINTER_CHA  ;RX BUFFER POINTER FOR CHA
                        64
                        65      ;I/O PORT ASSIGNMENTS
                        66
                        67      ;CHANNEL A PORT ASSIGNMENTS
                        68
0000                    69      DATA_PORT_CHA           EQU     0                       ;DATA I/O PORT
0002                    70      COMMAND_PORT_CHA        EQU     2                       ;COMMAND PORT
0002                    71      STATUS_PORT_CHA         EQU     COMMAND_PORT_CHA        ;STATUS PORT
                        72
                        73      ;CHANNEL B PORT ASSIGNMENTS
                        74
0004                    75      DATA_PORT_CHB           EQU     4                       ;DATA I/O PORT
0006                    76      COMMAND_PORT_CHB        EQU     6                       ;COMMAND PORT
0006                    77      STATUS_PORT_CHB         EQU     COMMAND_PORT_CHB        ;STATUS PORT
                        78
                        79      ;MISC  SYSTEM EQUATES
                        80
0D00                    81      CR_CHR          EQU     0DH     ;ASCII CR CHARACTER CODE
0200                    82      INT_TABLE_BASE  EQU     200H    ;INT  VECTOR BASE ADDRESS
0500                    83      CODE_START      EQU     500H    ;START LOCATION FOR CODE
                        84
                        85 +1   $EJECT
                        86
                        87      ;RAM ASSSIGNMENTS FOR DATA SEGMENT
                        88
----                    89              DATA    SEGMENT
                        90
```

210311-25

```
MCS-86 MACRO ASSEMBLER   ASYNCB


LOC  OBJ              LINE    SOURCE

                        91    ;VECTOR INTERRUPT TABLE - ASSUME INITIAL 8274 INTERRUPT
                        92    ;VECTOR IS NUMBER 80 (0200H)  FOR EACH VECTOR. THE TABLE
                        93    ;CONTAINS START LOCATION AND CODE SEGMENT REGISTER VALUE
                        94    ;THE TABLE IS LOADED FROM PROM
                        95
0200                    96            ORG     INT_TABLE_BASE
                        97
0200 0000               98    TX_VECTOR_CHB   DW      0       ;TX INTERRUPT VECTOR FOR CHB
0202 0000               99    TX_CS_CHB       DW      0
                       100
0204 0000              101    STS_VECTOR_CHB  DW      0       ;STATUS INTERRUPT VECTOR FOR CHB
0206 0000              102    STS_CS_CHB      DW      0
                       103
0208 0000              104    RX_VECTOR_CHB   DW      0       ;RX INTERRUPT VECTOR FOR CHB
020A 0000              105    RX_CS_CHB       DW      0
                       106
020C 0000              107    ERR_VECTOR_CHB  DW      0       ;ERROR INTERRUPT VECTOR FOR CHB
020E 0000              108    ERR_CS_CHB      DW      0
                       109
0210 0000              110    TX_VECTOR_CHA   DW      0       ;TX INTERRUPT VECTOR FOR CHA
0212 0000              111    TX_CS_CHA       DW      0
                       112
0214 0000              113    STS_VECTOR_CHA  DW      0       ;STATUS INTERRUPT VECTOR FOR CHA
0216 0000              114    STS_CS_CHA      DW      0
                       115
0218 0000              116    RX_VECTOR_CHA   DW      0       ;RX INTERRUPT VECTOR FOR CHA
021A 0000              117    RX_CS_CHA       DW      0
                       118
021C 0000              119    ERR_VECTOR_CHA  DW      0       ;ERROR INTERRUPT VECTOR FOR CHA
021E 0000              120    ERR_CS_CHA      DW      0
                       121
                       122    ;MISC RAM LOCATIONS FOR CHANNEL STATUS AND POINTERS
                       123
                       124    ;CHANNEL B POINTERS AND STATUS
                       125
0220 0000              126    TX_POINTER_CHB  DW      0       ;TX BUFFER POINTER FOR CHB
0222 0000              127    TX_LENGTH_CHB   DW      0       ;TX BUFFER LENGTH FOR CHB
0224 0000              128    RX_POINTER_CHB  DW      0       ;RX BUFFER POINTER FOR CHB
0226 0000              129    RX_COUNT_CHB    DW      0       ;RX LENGTH COUNTER FOR CHB
0228 00                130    TX_EMPTY_CHB    DB      0       ;TX DONE FLAG
0229 00                131    RX_READY_CHB    DB      0       ;READY FLAG (1 IF CR_CHR RECEIVED, ELSE 0)
022A 00                132    STATUS_MSG_CHB  DB      0       ;STATUS CHANGE MESSAGE
022B 00                133    ERROR_MSG_CHB   DB      0       ;ERROR STATUS LOCATION (0 IF NO ERROR)
                       134
                       135    ;CHANNEL A POINTERS AND STATUS
                       136
022C 0000              137    TX_POINTER_CHA  DW      0       ;TX BUFFER POINTER FOR CHA
022E 0000              138    TX_LENGTH_CHA   DW      0       ;TX BUFFER LENGTH FOR CHA
0230 0000              139    RX_POINTER_CHA  DW      0       ;RX BUFFER POINTER FOR CHA
0232 0000              140    RX_COUNT_CHA    DW      0       ;RX LENGTH COUNTER FOR CHA
0234 00                141    TX_EMPTY_CHA    DB      0       ;TX DONE FLAG
0235 00                142    RX_READY_CHA    DB      0       ;READY FLAG (1 IF CR_CHR RECEIVED, ELSE 0)
0236 00                143    STATUS_MSG_CHA  DB      0       ;STATUS CHANGE MESSAGE
0237 00                144    ERROR_MSG_CHA   DB      0       ;ERROR STATUS LOCATION (0 IF NO ERROR)
                       145
----                   146            DATA    ENDS
                       147
                       148 +1  $EJECT
```

210311-26

2

```
MCS-86 MACRO ASSEMBLER   ASYNCB


LOC  OBJ              LINE    SOURCE

                      149
----                  150           ABC     SEGMENT
                      151           ASSUME  CS:ABC,DS:DATA,SS:DATA
0500                  152           ORG     CODE_START
                      153
                      154     ;*************************************************
                      155     ;*                                               *
                      156     ;*          PARAMETERS FOR CHANNEL INITIALIZATION *
                      157     ;*                                               *
                      158     ;*************************************************
                      159
                      160     ;CHANNEL B PARAMETERS
                      161
                      162           ;WR1 - INTERRUPT ON ALL RX CHR, VARIABLE INT VECTOR, TX INT ENABLE
0500 01               163     CMDSTRB DB      1,16H
0501 16
                      164           ;WR2 - INTERRUPT VECTOR
0502 02               165           DB      2,(INT_TABLE_BASE/4)
0503 80
                      166           ;WR3 - RX 8 BITS/CHR, RX DISABLE
0504 03               167           DB      3,0C0H
0505 C0
                      168           ;WR4 - X16 CLOCK, 2 STOP BITS, NO PARITY
0506 04               169           DB      4,4CH
0507 4C
                      170           ;WR5 - DTR ACTIVE, TX 8 BITS/CHR, TX ENABLE, RTS ACTIVE
0508 05               171           DB      5,0EAH
0509 EA
                      172           ;WR6 AND WR7 NOT REQUIRED FOR ASYNC
050A 00               173           DB      0,0
050B 00
                      174
                      175     ;CHANNEL A PARAMETERS
                      176
                      177           ;WR1 - INTERRUPT ON ALL RX CHR, TX INT ENABLE
050C 01               178     CMDSTRA DB      1,12H
050D 12
                      179           ;WR2 - VECTORED INTERRUPT FOR 8086
050E 02               180           DB      2,38H
050F 38
                      181           ;WR3 - RX 8 BITS/CHR, RX DISABLE
0510 03               182           DB      3,0C0H
0511 C0
                      183           ;WR4 - X16 CLOCK, 2 STOP BITS, NO PARITY
0512 04               184           DB      4,4CH
0513 4C
                      185           ;WR5 - DTR ACTIVE, TX 8 BITS/CHR, TX ENABLE, RTS ACTIVE
0514 05               186           DB      5,0EAH
0515 EA
                      187           ;WR6 AND WR7 NOT REQUIRED FOR ASYNC
0516 00               188           DB      0,0
0517 00
                      189
                      190 +1  $EJECT
```

210311-27

```
MCS-86 MACRO ASSEMBLER   ASYNCB

LOC  OBJ                 LINE    SOURCE

                          191
                          192    ;START OF COMMAND ROUTINES
                          193
                          194    ;*************************************************
                          195    ;*                                               *
                          196    ;*     INITIALIZATION COMMAND FOR THE 8274 - THE 8274        *
                          197    ;*     IS SETUP ACCORDING TO THE PARAMETERS STORED IN        *
                          198    ;*     PROM ABOVE STARTING AT CMSTRB FOR CHANNEL B AND       *
                          199    ;*     CMSTRA FOR CHANNEL A                      *
                          200    ;*                                               *
                          201    ;*************************************************
                          202
0518                      203    INITIAL_8274:
                          204            ;COPY INTERRUPT VECTOR IP AND CS VALUES FROM PROM TO RAM
0518 C70600020806         205            MOV     TX_VECTOR_CHB, OFFSET XMTINB    ;TX DATA VECTOR CHB
051E 8C0E0202             206            MOV     TX_CS_CHB, CS
0522 C70604023506         207            MOV     STS_VECTOR_CHB, OFFSET STAINB   ;STATUS VECTOR CHB
0528 8C0E0602             208            MOV     STS_CS_CHB, CS
052C C70608024906         209            MOV     RX_VECTOR_CHB, OFFSET RCVINB    ;RX DATA VECTOR CHB
0532 8C0E0A02             210            MOV     RX_CS_CHB, CS
0536 C7060C027706         211            MOV     ERR_VECTOR_CHB, OFFSET ERRINB   ;ERROR VECTOR CHB
053C 8C0E0A02             212            MOV     RX_CS_CHB, CS
0540 C70610028C06         213            MOV     TX_VECTOR_CHA, OFFSET XMTINA    ;TX DATA VECTOR CHA
0546 8C0E1202             214            MOV     TX_CS_CHA, CS
054A C7061402B906         215            MOV     STS_VECTOR_CHA, OFFSET STAINA   ;STATUS VECTOR CHA
0550 8C0E1602             216            MOV     STS_CS_CHA, CS
0554 C7061802CD06         217            MOV     RX_VECTOR_CHA, OFFSET RCVINA    ;RX DATA VECTOR CHA
055A 8C0E1A02             218            MOV     RX_CS_CHA, CS
055E C7061C02FB06         219            MOV     ERR_VECTOR_CHA, OFFSET ERRINA   ;ERROR VECTOR CHA
0564 8C0E1E02             220            MOV     ERR_CS_CHA, CS
                          221
                          222    ;COPY SETUP TABLE PARAMETERS INTO 8274
                          223
0568 BF0005               224            MOV     DI, OFFSET CMDSTRB      ;INITIALIZE CHB
056B BA0600               225            MOV     DX, COMMAND_PORT_CHB
056E E82E00               226            CALL    SETUP                   ;COPY CHB PARAMETERS
0571 BF0C05               227            MOV     DI, OFFSET CMDSTRA      ;INITIALIZE CHA
0574 BA0200               228            MOV     DX, COMMAND_PORT_CHA
0577 E82500               229            CALL    SETUP                   ;COPY CHA PARAMETERS
                          230
                          231    ;INITIALIZE STATUS BYTES AND FLAGS
                          232
057A B80000               233            MOV     AX, 0
057D A22B02               234            MOV     ERROR_MSG_CHB, AL       ;CLEAR ERROR FLAG CHB
0580 A23702               235            MOV     ERROR_MSG_CHA, AL       ;CLEAR ERROR FLAG CHA
0583 A22A02               236            MOV     STATUS_MSG_CHB, AL      ;CLEAR STATUS FLAG CHB
0586 A23602               237            MOV     STATUS_MSG_CHA, AL      ;CLEAR STATUS FLAG CHA
0589 A32602               238            MOV     RX_COUNT_CHB, AX        ;CLEAR RX COUNTER CHB
058C A33202               239            MOV     RX_COUNT_CHA, AX        ;CLEAR RX COUNTER CHA
058F B001                 240            MOV     AL, 1
0591 A22902               241            MOV     RX_READY_CHB, AL        ;SET RX DONE FLAG CHB
0594 A23502               242            MOV     RX_READY_CHA, AL        ;SET RX DONE FLAG CHA
0597 A22802               243            MOV     TX_EMPTY_CHB, AL        ;SET TX DONE FLAG CHB
059A A23402               244            MOV     TX_EMPTY_CHA, AL        ;SET TX DONE FLAG CHA
059D FB                   245            STI                             ;ENABLE INTERRUPTS
059E C3                   246            RET                             ;RETURN - DONE WITH SETUP
                          247
059F 8A05                 248    SETUP:  MOV     AL, [DI]                ;PARAMETER COPYING ROUTINE
05A1 3C00                 249            CMP     AL, 0
05A3 7404                 250            JE      DONE
```

210311-28

```
LOC  OBJ              LINE    SOURCE

05A5 EE                251            OUT     DX, AL                  ;OUTPUT PARAMETER
05A6 47                252            INC     DI                      ;POINT AT NEXT PARAMETER
05A7 EBF6              253            JMP     SETUP                   ;GO LOAD IT
05A9 C3                254    DONE:   RET                             ;DONE - SO RETURN
                       255
                       256 +1 $EJECT
                       257
                       258    ;*************************************************************
                       259    ;*                                                           *
                       260    ;*    TX CHANNEL B COMMAND ROUTINE - ROUTINE IS CALLED TO    *
                       261    ;*    TRANSMIT A BUFFER.  THE BUFFER STARTING ADDRESS,       *
                       262    ;*    TX_POINTER_CHB, AND THE BUFFER LENGTH, TX_LENGTH_CHB,  *
                       263    ;*    MUST BE INITIALIZED BY THE CALLING PROGRAM             *
                       264    ;*    BOTH ITEMS ARE WORD VARIABLES                          *
                       265    ;*                                                           *
                       266    ;*************************************************************
                       267
05AA                   268    TX_COMMAND_CHB:
05AA 50                269            PUSH    AX              ;SAVE REGISTERS
05AB 57                270            PUSH    DI
05AC 52                271            PUSH    DX
05AD C606280200        272            MOV     TX_EMPTY_CHB, 0 ;CLEAR EMPTY FLAG
05B2 BA0400            273            MOV     DX, DATA_PORT_CHB       ;SETUP PORT POINTER
05B5 8B3E2002          274            MOV     DI, TX_POINTER_CHB      ;GET TX BUFFER POINTER CHB
05B9 8A05              275            MOV     AL, [DI]        ;GET FIRST CHARACTER TO TX
05BB EE                276            OUT     DX, AL          ;OUTPUT IT TO 8274 TO GET IT STARTED
05BC 5A                277            POP     DX
05BD 5F                278            POP     DI
05BE 58                279            POP     AX
05BF C3                280            RET                     ;RETURN
                       281
                       282    ;*************************************************************
                       283    ;*                                                           *
                       284    ;*    TX CHANNEL A COMMAND ROUTINE - ROUTINE IS CALLED TO    *
                       285    ;*    TRANSMIT A BUFFER.  THE BUFFER STARTING ADDRESS,       *
                       286    ;*    TX_POINTER_CHA, AND THE BUFFER LENGTH, TX_LENGTH_CHA,  *
                       287    ;*    MUST BE INITIALIZED BY THE CALLING PROGRAM             *
                       288    ;*    BOTH ITEMS ARE WORD VARIABLES                          *
                       289    ;*                                                           *
                       290    ;*************************************************************
                       291
05C0                   292    TX_COMMAND_CHA:
05C0 50                293            PUSH    AX              ;SAVE REGISTERS
05C1 57                294            PUSH    DI
05C2 52                295            PUSH    DX
05C3 C606340200        296            MOV     TX_EMPTY_CHA, 0 ;CLEAR EMPTY FLAG
05C8 BA0000            297            MOV     DX, DATA_PORT_CHA       ;SETUP PORT POINTER
05CB 8B3E2C02          298            MOV     DI, TX_POINTER_CHA      ;GET TX BUFFER POINTER CHA
05CF 8A05              299            MOV     AL, [DI]        ;GET FIRST CHARACTER TO TX
05D1 EE                300            OUT     DX, AL          ;OUTPUT IT TO 8274 TO GET IT STARTED
05D2 5A                301            POP     DX
05D3 5F                302            POP     DI
05D4 58                303            POP     AX
05D5 C3                304            RET                     ;RETURN
                       305
                       306    ;*************************************************************
                       307    ;*                                                           *
                       308    ;*    RX COMMAND FOR CHANNEL B - THE CALLING ROUTINE MUST    *
                       309    ;*    INITIALIZE RX_POINTER_CHB TO POINT AT THE RECEIVE      *
                       310    ;*    BUFFER BEFORE CALLING THIS ROUTINE                     *
```

210311-29

```
MCS-86 MACRO ASSEMBLER   ASYNCB

LOC  OBJ                 LINE    SOURCE

                          311    ;*                                                  *
                          312    ;****************************************************
                          313
05D6                      314    RX_COMMAND_CHB
05D6 50                   315           PUSH   AX              ;SAVE REGISTERS
05D7 52                   316           PUSH   DX
05D8 C606290200           317           MOV    RX_READY_CHB, 0 ;CLEAR RX READY FLAG
05DD C706260200000        318           MOV    RX_COUNT_CHB, 0 ;CLEAR RX COUNTER
05E3 BA0600               319           MOV    DX, COMMAND_PORT_CHB    ;POINT AT COMMAND PORT
05E6 B003                 320           MOV    AL, 3           ;SET UP FOR WR3
05E8 EE                   321           OUT    DX, AL
05E9 B0C1                 322           MOV    AL, 0C1H        ;WR3 = 8 BITS/CHR, ENABLE RX
05EB EE                   323           OUT    DX, AL
05EC 5A                   324           POP    DX
05ED 58                   325           POP    AX
05EE C3                   326           RET                    ;RETURN
                          327
                          328    ;****************************************************
                          329    ;*                                                  *
                          330    ;*    RX COMMAND FOR CHANNEL A - THE CALLING ROUTINE MUST   *
                          331    ;*    INITIALIZE RX_POINTER_CHA TO POINT AT THE RECEIVE     *
                          332    ;*    BUFFER BEFORE CALLING THIS ROUTINE            *
                          333    ;*                                                  *
                          334    ;****************************************************
                          335
05EF                      336    RX_COMMAND_CHA
05EF 50                   337           PUSH   AX              ;SAVE REGISTERS
05F0 52                   338           PUSH   DX
05F1 C606350200           339           MOV    RX_READY_CHA, 0 ;CLEAR RX READY FLAG
05F6 C706320200000        340           MOV    RX_COUNT_CHA, 0 ;CLEAR RX COUNTER
05FC BA0200               341           MOV    DX, COMMAND_PORT_CHA    ;POINT AT COMMAND PORT
05FF B003                 342           MOV    AL, 3           ;SET UP FOR WR3
0601 EE                   343           OUT    DX, AL
0602 B0C1                 344           MOV    AL, 0C1H        ;WR3 = 8 BITS/CHR, ENABLE RX
0604 EE                   345           OUT    DX, AL
0605 5A                   346           POP    DX
0606 58                   347           POP    AX
0607 C3                   348           RET                    ;RETURN
                          349
                          350 +1 $EJECT
                          351
                          352    ;****************************************************
                          353    ;*                                                  *
                          354    ;*          START OF INTERRUPT SERVICE ROUTINES     *
                          355    ;*                                                  *
                          356    ;****************************************************
                          357
                          358    ;CHANNEL B TRANSMIT DATA SERVICE ROUTINE
                          359
0608 52                   360    XMTINB: PUSH   DX              ;SAVE REGISTERS
0609 57                   361           PUSH   DI
060A 50                   362           PUSH   AX
060B E88201               363           CALL   EOI             ;SEND EOI COMMAND TO 8274
060E FF062002             364           INC    TX_POINTER_CHB  ;POINT TO NEXT CHARACTER
0612 FF0E2202             365           DEC    TX_LENGTH_CHB   ;DEC LENGTH COUNTER
0616 740E                 366           JE     XIB             ;TEST IF DONE
0618 BA0400               367           MOV    DX, DATA_PORT_CHB       ;NOT DONE - GET NEXT CHARACTER
061B 8B3E2002             368           MOV    DI, TX_POINTER_CHB
061F 8A05                 369           MOV    AL, [DI]        ;PUT CHARACTER IN AL
0621 EE                   370           OUT    DX, AL          ;OUTPUT IT TO 8274
```

210311-30

```
MCS-86 MACRO ASSEMBLER   ASYNCB

LOC  OBJ              LINE    SOURCE

0622 58               371           POP   AX              ;RESTORE REGISTERS
0623 5F               372           POP   DI
0624 5A               373           POP   DX
0625 CF               374           IRET                  ;RETURN TO FOREGROUND
0626 BA0600           375     XIB:  MOV   DX, COMMAND_PORT_CHB    ;ALL CHARACTERS HAVE BEEN SEND
0629 B028             376           MOV   AL, 28H         ;RESET TRANSMITTER INTERRUPT PENDING
062B EE               377           OUT   DX, AL
062C C606280201       378           MOV   TX_EMPTY_CHB, 1 ;DONE - SO SET TX EMPTY FLAG CHB
0631 58               379           POP   AX              ;RESTORE REGISTERS
0632 5F               380           POP   DI
0633 5A               381           POP   DX
0634 CF               382           IRET                  ;RETURN TO FOREGROUND
                      383
                      384     ;CHANNEL B STATUS CHANGE SERVICE ROUTINE
                      385
0635 52               386     STAINB: PUSH  DX            ;SAVE REGISTERS
0636 57               387           PUSH  DI
0637 50               388           PUSH  AX
0638 E8D500           389           CALL  EOI             ;SEND EOI COMMAND TO 8274
063B BA0600           390           MOV   DX, COMMAND_PORT_CHB
063E EC               391           IN    AL, DX          ;READ RR0
063F A22A02           392           MOV   STATUS_MSG_CHB, AL    ;PUT RR0 IN STATUS MESSAGE
0642 B010             393           MOV   AL, 10H         ;SEND RESET STATUS INT COMMAND TO 8274
0644 EE               394           OUT   DX, AL
0645 58               395           POP   AX              ;RESTORE REGISTERS
0646 5F               396           POP   DI
0647 5A               397           POP   DX
0648 CF               398           IRET
                      399
                      400     ;CHANNEL B RECEIVED DATA SERVICE ROUTINE
                      401
0649 52               402     RCVINB: PUSH  DX            ;SAVE REGISTERS
064A 57               403           PUSH  DI
064B 50               404           PUSH  AX
064C E8C100           405           CALL  EOI             ;SEND EOI COMMAND TO 8274
064F 8B3E2402         406           MOV   DI, RX_POINTER_CHB    ;GET RX CHB BUFFER POINTER
0653 BA0400           407           MOV   DX, DATA_PORT_CHB
0656 EC               408           IN    AL, DX          ;READ CHARACTER
0657 8805             409           MOV   [DI], AL        ;STORE IN BUFFER
0659 FF062402         410           INC   RX_POINTER_CHB  ;BUMP THE BUFFER POINTER
065D FF062602         411           INC   RX_COUNT_CHB    ;BUMP THE COUNTER
0661 3C0D             412           CMP   AL, CR_CHR      ;TEST IF LAST CHARACTER TO BE RECEIVED?
0663 750E             413           JNE   RIB
0665 C606290201       414           MOV   RX_READY_CHB, 1 ;YES, SET READY FLAG
066A BA0600           415           MOV   DX, COMMAND_PORT_CHB    ;POINT AT COMMAND PORT
066D B003             416           MOV   AL, 3           ;POINT AT WR3
066F EE               417           OUT   DX, AL
0670 B0C0             418           MOV   AL, 0C0H        ;DISABLE RX
0672 EE               419           OUT   DX, AL
0673 58               420     RIB:  POP   AX              ;EITHER WAY, RESTORE REGISTERS
0674 5F               421           POP   DI
0675 5A               422           POP   DX
0676 CF               423           IRET                  ;RETURN TO FOREGROUND
                      424
                      425     ;CHANNEL B ERROR SERVICE ROUTINE
                      426
0677 52               427     ERRINB: PUSH  DX            ;SAVE REGISTERS
0678 50               428           PUSH  AX
0679 E89400           429           CALL  EOI             ;SEND EOI COMMAND TO 8274
067C BA0600           430           MOV   DX, COMMAND_PORT_CHB
```

210311-31

```
MCS-86 MACRO ASSEMBLER   ASYNCB

LOC  OBJ              LINE    SOURCE

067F B001              431            MOV     AL, 1           ;POINT AT RR1
0681 EE                432            OUT     DX, AL
0682 EC                433            IN      AL, DX          ;READ RR1
0683 A22B02            434            MOV     ERROR_MSG_CHB, AL       ;SAVE IT IN ERROR FLAG
0686 B030              435            MOV     AL, 30H         ;SEND RESET ERROR COMMAND TO 8274
0688 EE                436            OUT     DX, AL
0689 58                437            POP     AX              ;RESTORE REGISTERS
068A 5A                438            POP     DX
068B CF                439            IRET                    ;RETURN TO FOREGROUND
                       440
                       441    ;CHANNEL A TRANSMIT DATA SERVICE ROUTINE
                       442
068C 52                443    XMTINA: PUSH    DX              ;SAVE REGISTERS
068D 57                444            PUSH    DI
068E 50                445            PUSH    AX
068F E87E00            446            CALL    EOI             ;SEND EOI COMMAND TO 8274
0692 FF062C02          447            INC     TX_POINTER_CHA  ;POINT TO NEXT CHARACTER
0696 FF0E2E02          448            DEC     TX_LENGTH_CHA   ;DEC LENGTH COUNTER
069A 740E              449            JE      XIA             ;TEST IF DONE
069C BA0000            450            MOV     DX, DATA_PORT_CHA       ;NOT DONE - GET NEXT CHARACTER
069F 8B3E2C02          451            MOV     DI, TX_POINTER_CHA
06A3 8A05              452            MOV     AL, [DI]        ;PUT CHARACTER IN AL
06A5 EE                453            OUT     DX, AL          ;OUTPUT IT TO 8274
06A6 58                454            POP     AX              ;RESTORE REGISTERS
06A7 5F                455            POP     DI
06A8 5A                456            POP     DX
06A9 CF                457            IRET                    ;RETURN TO FOREGROUND
06AA BA0200            458    XIA:    MOV     DX, COMMAND_PORT_CHA    ;ALL CHARACTERS HAVE BEEN SEND
06AD B028              459            MOV     AL, 28H         ;RESET TRANSMITTER INTERRUPT PENDING
06AF EE                460            OUT     DX, AL
06B0 C606348201        461            MOV     TX_EMPTY_CHA, 1 ;DONE - SO SET TX EMPTY FLAG CHB
06B5 58                462            POP     AX              ;RESTORE REGISTERS
06B6 5F                463            POP     DI
06B7 5A                464            POP     DX
06B8 CF                465            IRET                    ;RETURN TO FOREGROUND
                       466
                       467    ;CHANNEL A STATUS CHANGE SERVICE ROUTINE
                       468
06B9 52                469    STAINA: PUSH    DX              ;SAVE REGISTERS
06BA 57                470            PUSH    DI
06BB 50                471            PUSH    AX
06BC E85100            472            CALL    EOI             ;SEND EOI COMMAND TO 8274
06BF BA0200            473            MOV     DX, COMMAND_PORT_CHA
06C2 EC                474            IN      AL, DX          ;READ RR0
06C3 A23602            475            MOV     STATUS_MSG_CHA, AL      ;PUT RR0 IN STATUS MESSAGE
06C6 B010              476            MOV     AL, 10H         ;SEND RESET STATUS INT COMMAND TO 8274
06C8 EE                477            OUT     DX, AL
06C9 58                478            POP     AX              ;RESTORE REGISTERS
06CA 5F                479            POP     DI
06CB 5A                480            POP     DX
06CC CF                481            IRET
                       482
                       483    ;CHANNEL A RECEIVED DATA SERVICE ROUTINE
                       484
06CD 52                485    RCVINA: PUSH    DX              ;SAVE REGISTERS
06CE 57                486            PUSH    DI
06CF 50                487            PUSH    AX
06D0 E83D00            488            CALL    EOI             ;SEND EOI COMMAND TO 8274
06D3 8B3E2802          489            MOV     DI, RX_POINTER_CHA      ;GET RX CHA BUFFER POINTER
06D7 BA0000            490            MOV     DX, DATA_PORT_CHA
```

210311-32

```
MCS-86 MACRO ASSEMBLER   ASYNCB


LOC  OBJ              LINE    SOURCE

06DA EC                491            IN      AL, DX          ;READ CHARACTER
06DB 8805              492            MOV     [DI], AL        ;STORE IN BUFFER
06DD FF063002          493            INC     RX_POINTER_CHA  ;BUMP THE BUFFER POINTER
06E1 FF063202          494            INC     RX_COUNT_CHA    ;BUMP THE COUNTER
06E5 3C0D              495            CMP     AL, CR_CHR      ;TEST IF LAST CHARACTER TO BE RECEIVED?
06E7 750E              496            JNE     RIA
06E9 C606350201        497            MOV     RX_READY_CHA, 1 ;YES, SET READY FLAG
06EE BA0200            498            MOV     DX, COMMAND_PORT_CHA    ;POINT AT COMMAND PORT
06F1 B003              499            MOV     AL, 3           ;POINT AT WR3
06F3 EE                500            OUT     DX, AL
06F4 B0C0              501            MOV     AL, 0C0H        ;DISABLE RX
06F6 EE                502            OUT     DX, AL
06F7 58                503    RIA:    POP     AX              ;EITHER WAY, RESTORE REGISTERS
06F8 5F                504            POP     DI
06F9 5A                505            POP     DX
06FA CF                506            IRET                    ;RETURN TO FOREGROUND
                       507
                       508    ;CHANNEL A ERROR SERVICE ROUTINE
                       509
06FB 52                510    ERRINA: PUSH    DX              ;SAVE REGISTERS
06FC 50                511            PUSH    AX
06FD E81000            512            CALL    EOI             ;SEND EOI COMMAND TO 8274
0700 BA0200            513            MOV     DX, COMMAND_PORT_CHA
0703 B001              514            MOV     AL, 1           ;POINT AT RR1
0705 EE                515            OUT     DX, AL
0706 EC                516            IN      AL, DX          ;READ RR1
0707 A23702            517            MOV     ERROR_MSG_CHA, AL       ;SAVE IT IN ERROR FLAG
070A B030              518            MOV     AL, 30H         ;SEND RESET ERROR COMMAND TO 8274
070C EE                519            OUT     DX, AL
070D 58                520            POP     AX              ;RESTORE REGISTERS
070E 5A                521            POP     DX
070F CF                522            IRET                    ;RETURN TO FOREGROUND
                       523
                       524     END-OF-INTERRUPT ROUTINE - SENDS EOI COMMAND TO 8274.
                       525       THIS COMMAND MUST ALWAYS TO ISSUED ON CHANNEL A.
                       526
0710 50                527    EOI:    PUSH    AX              ;SAVE REGISTERS
0711 52                528            PUSH    DX
0712 BA0200            529            MOV     DX, COMMAND_PORT_CHA    ;ALWAYS FOR CHANNEL A !!!
0715 B038              530            MOV     AL, 38H
0717 EE                531            OUT     DX, AL
0718 5A                532            POP     DX
0719 58                533            POP     AX
071A C3                534            RET
                       535
                       536    ;END OF CODE ROUTINE
                       537
----                   538    ABC     ENDS
                       539            END

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

210311-33

REFERENCES

1. 8274 Multiprotocol Serial Controller (MPSC) Data Sheet, Intel Corporation, California, 1980.
2. Basics of Data Communication, Electronics Book Series, McGraw-Hill, New York, 1976.
3. Telecommunications and the Computer, J. Martin, Prentice-Hall, New Jersey, 1976.
4. Technical Aspects of Data Communications, J. McNamara, DEC Press, Massachusetts, 1977.
5. Miscellaneous Data Communications Standards—EIA RS-232-C, EIA RS-422, EIA RS-423, EIA Standard Sales, Washington, D.C.

Synchronous Communication with the 8274 Multiple Protocol Serial Controller

SIKANDAR NAQVI
APPLICATION ENGINEER

Order Number: 210403-001

SYNCHRONOUS COMMUNICATION WITH THE 8274 MULTIPLE PROTOCOL SERIAL CONTROLLER

CONTENTS

	PAGE
INTRODUCTION	2-389
SYNCHRONOUS PROTOCOL OVERVIEW	2-389
BLOCK DIAGRAM	2-391
MPSC INTERRUPT STRUCTURE	2-394
MPSC MODES OF OPERATION	2-398
APPLICATION EXAMPLE	2-403
SPECIAL APPLICATIONS	2-408
PROGRAMMING HINTS	2-410
APPENDIX A: APPLICATION EXAMPLE SOFTWARE LISTINGS	2-411
APPENDIX B: MPSC READ/WRITE REGISTER DESCRIPTIONS	2-422
REFERENCES	2-425

INTRODUCTION

The INTEL 8274 is a Multi-Protocol Serial Controller, capable of handling both asynchronous and synchronous communication protocols. Its programmable features allow it to be configured in various operating modes, providing opimization to given data communication application.

This application note describes the features of the MPSC in Synchronous Communication applications only. It is strongly recommended that the reader read the 8274 Data Sheet and Application Note AP134 "Asynchronous Communication with the 8274 Multi-Protocol Serial Controller" before reading this Application Note. This Application note assumes that the reader is familiar with the basic structure of the MPSC, in terms of pin description, Read/Write registers and asynchronous communication with the 8274. Appendix A contains the software listings of the Application Example and Appendix B shows the MPSC Read/Write Registers for quick reference.

The first section of this application note presents an overview of the various synchronous protocols. The second section discusses the block diagram description of the MPSC. This is followed by the description of MPSC interrupt structure and mode of operation in the third and fourth sections. The fifth section describes a hardware/software example, using the INTEL single board computer iSBC88/45 as the hardware vehicle. The sixth section consists of some specialized applications of the MPSC. Finally, in section seven, some useful programming hints are summarized.

SYNCHRONOUS PROTOCOL OVERVIEW

This section presents an overview of various synchronous protocols. The contents of this section are fairly tutorial and may be skipped by the more knowledgeable reader.

Bit Oriented Protocols Overview

Bit oriented protocols have been defined to manage the flow of information on data communication links. One of the most widely known protocols is the one defined by the International Standards Organization: HDLC (High Level Data Link Control). The American Standards Association's protocol, ADCCP is similar to HDLC. CCITT Recommendation X.25 layer 2 is also an acceptable version of HDLC. Finally, IBM's SDLC (Synchronous Data Link Control) is also a subset of the HDLC.

In this section, we will concentrate most of our discussion on HDLC. Figure 1 shows a basic HDLC frame format.

A frame consists of five basic fields: Flag, Address, Control, Data and Error Detection. A frame is bounded by flags—opening and closing flags. An address field is 8 bits wide, extendable to 2 or more bytes. The control field is also 8 bits wide, extendable to two bytes. The data field or information field may be any number of bits. The data field may or may not be on an 8-bit boundary. A powerful error detection code called Frame Check Sequence contains the calculated CRC (Cycle Redundancy Code) for all the bits between the flags.

ZERO BIT INSERTION

The flag has a unique binary bit pattern: 7E HEX. To eliminate the possibility of the data field containing a 7E HEX pattern, a bit stuffing technique called Zero Bit Insertion is used. This technique specifies that during transmission, a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1's. This will ensure that no pattern of 0 1 1 1 1 1 1 0 is ever transmitted between flags. On the receiving side, after receiving the flag, the receiver hardware automatically deletes any 0 following five consecutive 1's. The 8274 performs zero bit insertion and deletion automatically in the SDLC/HDLC mode. The zero-bit stuffing ensures periodic transitions in the data stream. These transitions are necessary for a phase lock circuit, which may be used at the receiver end to generate a receive clock which is in phase to the received data. The inserted and deleted 0's are not included in the CRC checking. The *address* field is used to address a given secondary station. The *control* field contains the link-level control information which includes implied acknowledgement, supervisory commands and responses, etc. A more detailed discussion of higher level protocol functions is beyond the scope of this application note. Interested readers may refer to the references at the end of this application note..

Opening Flag Byte	Address* Field (A)	Control** Field (C)	Data Field	Frame Check Sequence	Closing Flag Byte

Figure 1. HDLC/SDLC Frame Format

*Extendable to 2 or More Bytes.
**Extendable to 2 Bytes.

NON-RETURN TO ZERO INVERTED (NRZI)

NRZI is a method of clock and data encoding that is well suited to the HDLC protocol. It allows HDLC protocols to be used with low cost asynchronous modems. NRZI coding is done at the transmitter to enable clock recovery from the data at the receiver terminal by using standard digital phase locked loop techniques. NRZI coding specifies that the signal condition does not change for transmitting a 1, while a 0 causes a change of state. NRZI coding ensures that an active data line will have transition at least every 5-bit times (recall Zero Bit Insertion), while contiguous 0's will cause a change of state. Thus, ZBI and NRZI encoding makes it possible for a phase lock circuit at the receiver end to derive a receive clock (from received data) which is synchronized to the received data and at the same time ensure data transparency.

Byte Synchronous Communication

As the name implies, Byte Synchronous Communication is a synchronous communication protocol which means that the transmitting station is synchronized to the receiving station through the recognition of a special sync character or characters. Two examples of Byte Synchronous protocol are the IBM Bisync and Mono- ... more detailed discussion are directed to the references listed at the end of this Application Note.

As shown in Figure 2, each message is preceded by two sync characters. Since the sync characters are defined at the beginning of the message only, the transmitter must insert fill characters (sync) in order to maintain synchronization with the receiver when no data is being transmitted.

TRANSPARENT TRANSMISSION

Bisync protocol requires special control characters to maintain the communication link over the line. If the data is EBCDIC encoded, then transparency is ensured by the fact that the field will not contain any of the bisync control characters. However, if data does not conform to standard character encoding techniques, transparency in bisync is achieved by inserting a special character DLE (Data Link Escape) before and after a string of characters which are to be transmitted transparently. This ensures that any data characters which match any of the special characters are not confused for special characters. An example of a transparent block is shown in Figure 3.

In a transparent mode, it is required that the CRC (BCC) is not performed on special characters. Later on, we will show how the 8274 can be used to achieve transparent transmission in Bisync mode.

SYNC	SYNC	SOH	HEADER	STX TEXT	ETX OR ETB	CRC 1	CRC 2

Figure 2. Bisync Message Format

DLE	STX	TRANSPARENT TRANSMISSION	DLE	ETX	BCC

Enter transparent mode — return to normal mode

Figure 3. Bisync Transparent Format

BLOCK DIAGRAM

This section discusses the block diagram view of the 8274. The CPU interface and serial interface is discussed separately. This will be followed by a hardware example in the fifth section, which will show how to interface the 8274 with the Intel CPU 8088. The 8274 block diagram is shown in Figure 4.

CPU Interface

The CPU interface to the system interface logic block utilizes the A0, A1, $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ inputs to communicate with the internal registers of the 8274. Figure 5 shows the address of the internal registers. The DMA interface is achieved by utilizing DMA request lines for each channel: $TxDRQ_A$, $TxDRQ_B$, $RxDRQ_A$, $RxDRQ_B$. Note that $TxDRQ_B$ and $RxDRQ_B$ become $\overline{IPO}$ and $\overline{IPI}$ respectively in non-DMA mode. $\overline{IPI}$ is the Interrupt Priority Input and $\overline{IPO}$ is the Interrupt Priority Output. These two pins can be used for connecting multiple MPSCs in a daisy chain. If the Wait Mode is programmed, then $TxRDQ_A$ and $RxDRQ_A$ pins become RDY_B and RDY_A pins. These pins can be wire-OR'ed and are usually hooked up to the CPU RDY line to synchronize the CPU for block transfers. The $\overline{INT}$ pin is activated whenever the MPSC requires CPU attention. The $\overline{INTA}$ may be used to utilize the powerful vectored mode feature of the 8274. Detailed discussion on these subjects will be done later in this Application Note. The $\overline{RESET}$ pin may be used for hardware reset while the clock is required to click the internal logic on the MPSC.

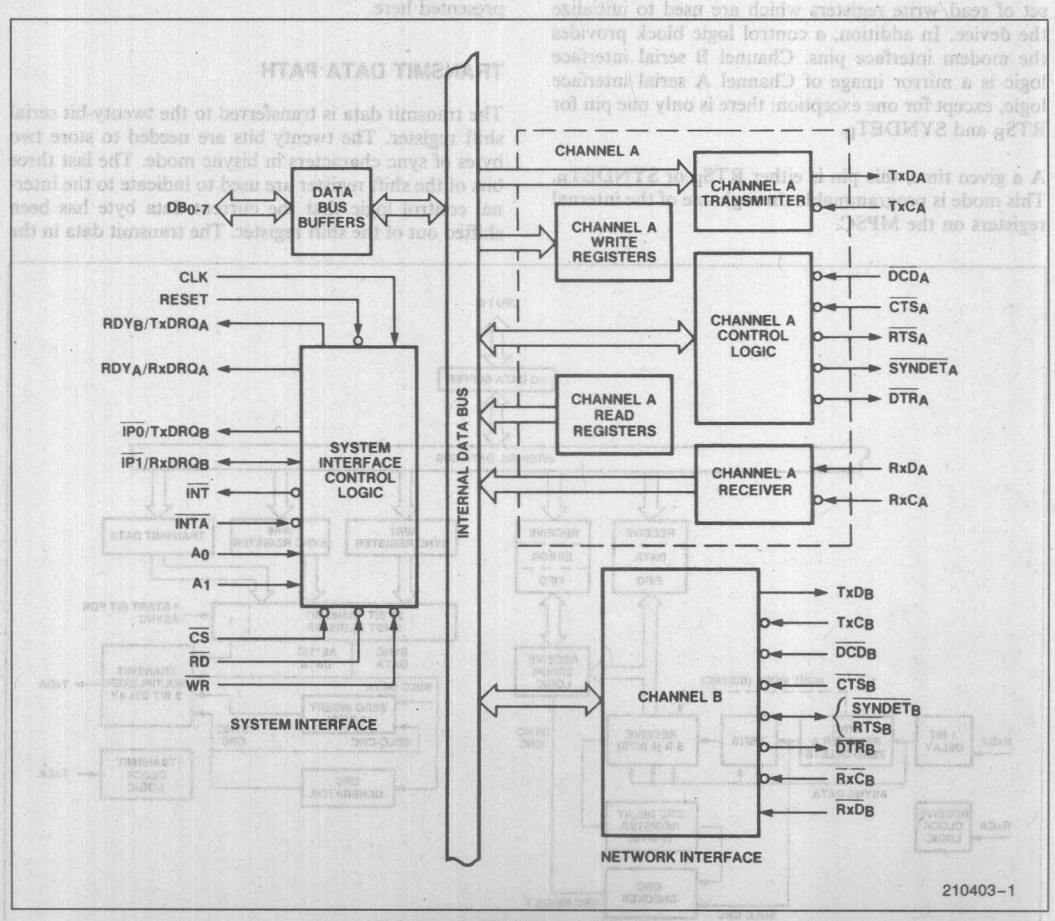


Figure 4. 8274 Block Diagram

$\overline{CS}$	A1	A0	Read Operation	Write Operation
0	0	0	CHA DATA READ	CHA DATA WRITE
0	1	0	CHA STATUS REGISTER (RR0,RR1)	CHA COMMAND/PARAMETER (WR0–WR7)
0	0	1	CHB DATA READ	CHB DATA WRITE
0	1	1	CHB STATUS REGISTER (RR0,RR1,RR2)	CHB COMMAND/PARAMETER (WR0–WR7)
1	X	X	HIGH Z	HIGH Z

Figure 5. Bus Interface

Serial Interface

On the serial side, there are two completely independent channels: Channel A and Channel B. Each channel consists of a transmitter block, receiver block and a set of read/write registers which are used to initialize the device. In addition, a control logic block provides the modem interface pins. Channel B serial interface logic is a mirror image of Channel A serial interface logic, except for one exception: there is only one pin for RTS_B and $SYNDET_B$.

A a given time, this pin is either RTS_B or $SYNDET_B$. This mode is programmable through one of the internal registers on the MPSC.

Transmit and Receive Data Path

Figure 6 shows a block diagram for transmit and receive data path. Without describing each block on the diagram, a brief discussion of the block diagram will be presented here.

TRANSMIT DATA PATH

The transmit data is transferred to the twenty-bit serial shift register. The twenty bits are needed to store two bytes of sync characters in bisync mode. The last three bits of the shift register are used to indicate to the internal control logic that the current data byte has been shifted out of the shift register. The transmit data in the

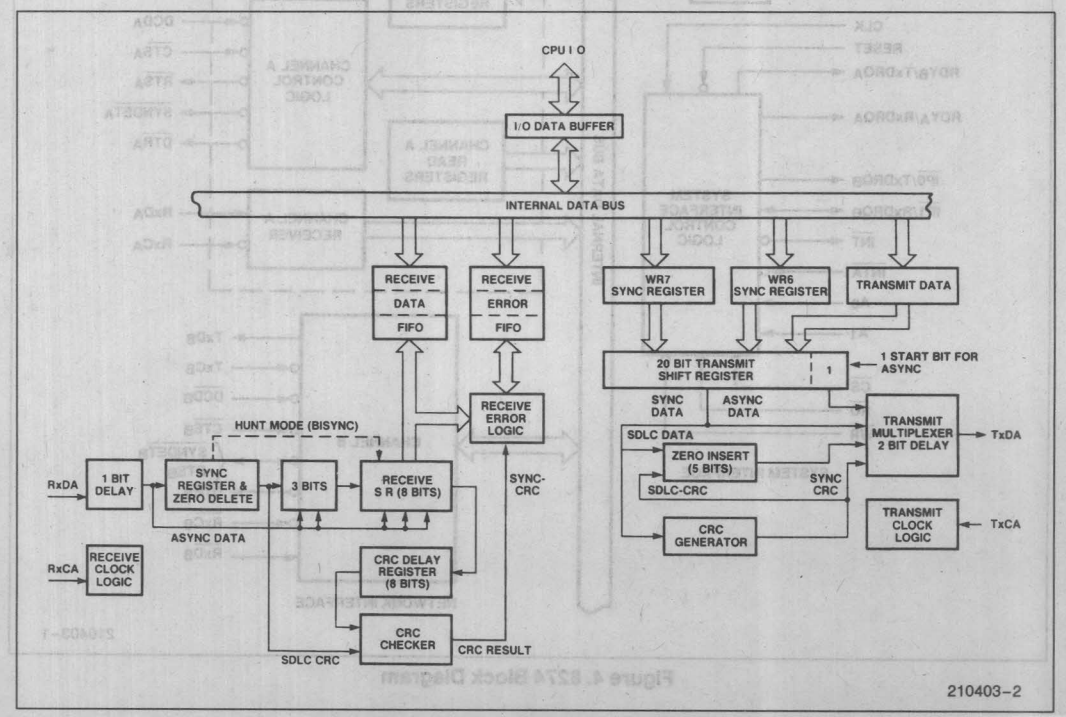


Figure 6. Transmit and Receive Data Path

transmit shift register is shifted out through a two bit delay onto the TxData line. This two bit delay is used to synchronize the internal shift clock with the external transmit clock. The data in the shift register is also presented to zero bit insertion logic which inserts a zero after sensing five contiguous ones in the data stream. In parallel to all this activity, the CRC-generator is computing CRC on the transmitted data and appends the frame with CRC bytes at the end of the data transmission.

RECEIVE DATA PATH

The received data is passed through a one bit delay before it is presented for flag/sync comparison. In bisync mode, after the synchronization is achieved, the incoming data bypasses the sync register and enters directly into the three bit buffer on its way to receive shift register. In SDLC mode, the incoming data always passes through the sync register where the data pattern is continuously monitored for contiguous ones for the

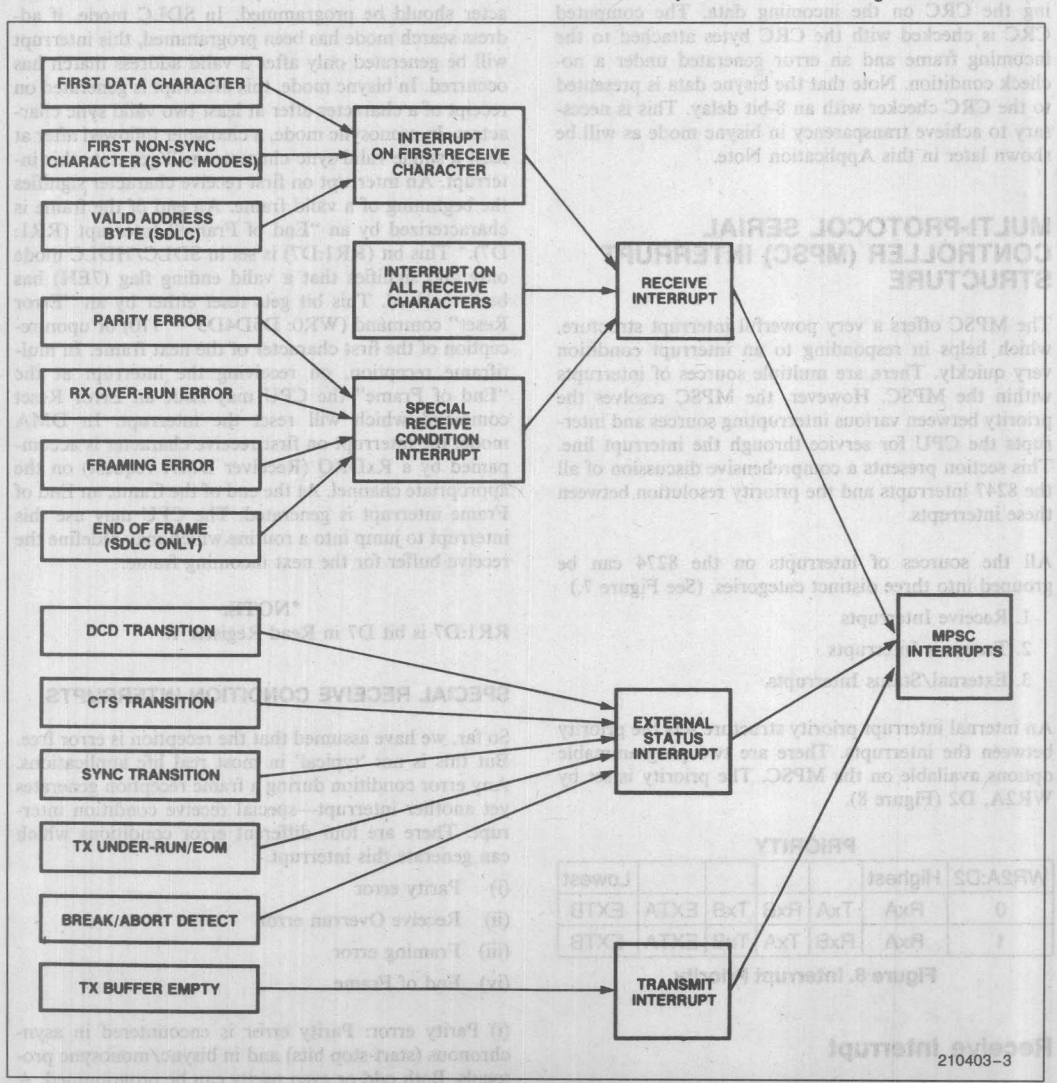


Figure 7. MPSC Interrupt Structure

shown later in this Application Note.

MULTI-PROTOCOL SERIAL CONTROLLER (MPSC) INTERRUPT STRUCTURE

The MPSC offers a very powerful interrupt structure, which helps in responding to an interrupt condition very quickly. There are multiple sources of interrupts within the MPSC. However, the MPSC resolves the priority between various interrupting sources and interrupts the CPU for service through the interrupt line. This section presents a comprehensive discussion of all the 8247 interrupts and the priority resolution between these interrupts.

All the sources of interrupts on the 8274 can be grouped into three distinct categories. (See Figure 7.)

1. Receive Interrupts
2. Transmit Interrupts
3. External/Status Interrupts.

An internal interrupt priority structure sets the priority between the interrupts. There are two programmable options available on the MPSC. The priority is set by WR2A, D2 (Figure 8).

PRIORITY

WR2A:D2	Highest					Lowest
0	RxA	TxA	RxB	TxB	EXTA	EXTB
1	RxA	RxB	TxA	TxB	EXTA	EXTB

Figure 8. Interrupt Priority

Receive Interrupt

All receive interrupts may be categorized into two distinct groups: Receive Interrupt on Receive Character and Special Receive Condition Interrupts.

least a single valid sync character will generate this interrupt. An interrupt on first receive character signifies the beginning of a valid frame. An end of the frame is characterized by an "End of Frame" Interrupt (RR1: D7).* This bit (RR1:D7) is set in SDLC/HDLC mode only and signifies that a valid ending flag (7EH) has been received. This bit gets reset either by an "Error Reset" command (WR0: D5D4D3 = 110) or upon reception of the first character of the next frame. In multiframe reception, on receiving the interrupt at the "End of Frame" the CPU may issue an Error Reset command which will reset the interrupt. In DMA mode, the interrupt on first receive character is accompanied by a RxDRQ (Receiver DMA request) on the appropriate channel. At the end of the frame, an End of Frame interrupt is generated. The CPU may use this interrupt to jump into a routine which may redefine the receive buffer for the next incoming frame.

***NOTE:**
RR1:D7 is bit D7 in Read Register 1.

SPECIAL RECEIVE CONDITION INTERRUPTS

So far, we have assumed that the reception is error free. But this is not 'typical' in most real life applications. Any error condition during a frame reception generates yet another interrupt—special receive condition interrupt. There are four different error conditions which can generate this interrupt.

(i) Parity error
(ii) Receive Overrun error
(iii) Framing error
(iv) End of Frame

(i) Parity error: Parity error is encountered in asynchronous (start-stop bits) and in bisync/monosync protocols. Both odd or even parity can be programmed. A parity error in a received byte will generate a special receive condition interrupt and sets bit 4 in RR1.

(ii) Receive Overrun error: If the CPU or the DMA controller (in DMA mode) fails to read a received character within three byte times after the received character interrupt (or DMA request) was generated, the receiver buffer will overflow and this will generate a special receive condition interrupt and sets bit 5 in RR1.

(iii) Framing error: In asynchronous mode, a framing error will generate a special receive interrupt and set bit D6 in RR1. This bit is not latched and is updated on the next received character.

(iv) End of frame: This interrupt is encountered in SDLC/HDLC mode only. When the MPSC receives the closing flag, it generates the special receive condition interrupt and sets bit D7 in RR1.

All the special receive condition interrupts may be reset by issuing an Error Reset Command.

CRC Error: In SDLC/HDLC and synchronous modes, a CRC error is indicated by bit D6 in RR1. When used to check CRC error, this bit is normally set until a correct CRC match is obtained which resets this bit. After receiving a frame, the CPU must read this bit (RR1:D6) to determine if a valid CRC check had occurred. It may be noted that a CRC error does not generate an interrupt.

It may also be pointed out that in SDLC/HDLC mode, receive DMA requests are disabled by a special receive condition and can only be re-enabled by issuing an Error Reset Command.

Transmit Interrupt

A transmit buffer empty generates a transmit interrupt. This has been discussed earlier under "Transmit in Interrupt Mode" and it would be sufficient to note here that a transmit buffer empty interrupt is generated only when the transmit buffer gets empty—assuming it had a data character loaded into it earlier. This is why on starting a frame transmission, the first data character is loaded by the CPU without a transmit empty interrupt (or DMA request in DMA mode). After this character is loaded into the serial shift register, the buffer becomes empty, and an interrupt (or DMA request) is generated. This interrupt is reset by a "Reset Tx Interrupt/DMA Pending" command (WR0: D5 D4 D3 = 101).

External/Status Interrupt

Continuing our discussion on transmit interrupt, if the transmit buffer is empty and the transmit serial shift register also becomes empty (due to the data character shifted out of the MPSC), a transmit under-run interrupt will be generated. This interrupt may be reset by "Reset External/Status Interrupt" command (WR0: D5 D4 D3 = 101).

The External Status Interrupt can be caused by five different conditions:

(i) CD Transition
(ii) CTS Transition
(iii) Sync/Hunt Transition
(iv) Tx under-run/EOM condition
(v) Break/Abort Detection.

CD, CTS TRANSITION

Any transition on these inputs on the serial interface will generate an External/Status interrupt and set the corresponding bits in status register RR0. This interrupt will also be generated in DMA as well as in Wait Mode. In order to find out the state of the $\overline{CTS}$ or $\overline{CD}$ pins *before* the transition had occurred, RR0 must be read before issuing a Reset External/Status Command through WR0. A read of RR0 after the Reset External/Status Command will give the condition of $\overline{CTS}$ or $\overline{CD}$ pins *after* the transition had occurred. Note that bit D5 in RR0 gives the complement of the state of $\overline{CTS}$ pin while D3 in RR0 reflects the actual state of the $\overline{CD}$ pin.

SYNC HUNT TRANSITION

Any transition of the $\overline{SYNDET}$ input generates an interrupt. However, sync input has different functions in different modes and we shall discuss them individually.

SDLC Mode

In SDLC mode, the $\overline{SYNDET}$ pin is an output. Status register RR1, D4 contains the state of the $\overline{SYNDET}$ pin. The Enter Hunt Mode initially sets this bit in R0. An opening flag in a received SDLC frame resets this bit and generates an external status interrupt. Every time the receiver is enabled or the Enter Hunt Code Command is issued, an external status interrupt will be generated on receiving a valid flag followed by a valid address/data character. This interrupt may be reset by the "Reset External/Status Interrupt" command.

External SYNC Mode

The MPSC can be programmed into External Sync Mode by setting WR4, D5 D4 = 11. The $\overline{SYNDET}$ pin is an input in this case and must be held high until an external character synchronization is established. However, the External Sync mode is enabled by the Enter Hunt Mode control bit (WR3: D4). A high at the $\overline{SYNDET}$ pin holds the Sync/Hunt bit (RR0,D4) in the reset state. When external synchronization is established, $\overline{SYNDET}$ must be driven low on second rising

edge of RxC after the rising edge of RxC on which the last bit of sync character was received. This high to low transition sets the Sync/Hunt bit and generates an external/status interrupt, which must be reset by the Reset External/Status command. If the $\overline{\text{SYNDET}}$ input goes high again, another External Status Interrupt is generated, which may be cleared by Reset External/Status command.

Mono-Sync/Bisync Mode

$\overline{\text{SYNDET}}$ pin acts as an output in this case. The Enter Hunt Mode sets the Sync/Hunt bit in R0. Sync/Hunt bit is reset when the MPSC achieves character synchronization. This high to low transition will generate an external status interrupt. The $\overline{\text{SYNDET}}$ pin goes active every time a sync pattern is detected in the data stream. Once again, the external status interrupt may be reset by the Reset External/Status command.

Tx UNDER-RUN/END OF MESSAGE (EOM)

The transmitter logic includes a transmit buffer and a transmit serial shift register. The CPU loads the character into the transmit buffer which is transferred into the transmit shift register to be shifted out of the MPSC. If the transmit buffer gets empty, a transmit buffer empty interrupt is generated (as discussed earlier). However, if the transmit buffer gets empty *and* the serial shift register gets empty, a transmit under-run condition will be created. This generates an External Status Interrupt and the interrupt can be cleared by the Reset External Status command. The status register RR0, D6 bit is set when the transmitter under-runs. This bit plays an important role in controlling a transmit operation, as will be discussed later in this application note.

BREAK/ABORT DETECTION

In asynchronous mode, bit D7 in RR0 is set when a break condition is detected on the receive data line. This also generates an External/Status interrupt which may be reset by issuing a Reset External/Status Interrupt command to the MPSC. Bit D7 in RR0 is reset when the break condition is terminated on the receive data line and this causes another External/Status interrupt to ge generated. Again, a Reset External/Status Interrupt command will reset this interrupt and will enable the break detection logic to look for the next break sequence.

In SDLC Receive Mode, an Abort sequence (seven or more 1's) detection on the receive data line will generate an External/Status interrupt and set RR0,D7. A Reset External/Status command will clear this interrupt. However, a termination of the Abort sequence will generate another interrupt and set RR0,D7 again. Once again, it may be cleared by issuing Reset External/Status Command.

This concludes our discussion on External Status Interrupts.

Interrupt Priority Resolution

The internal interrupt priority between various interrupt sources is resolved by an internal priority logic circuit, according to the priority set in WR2A. We will now discuss the interrupt timings during the priority resolution. Figures 9 and 10 show the timing diagrams for vectored and non-vectored modes.

VECTORED MODE

We shall assume that the MPSC accepted an internal request for an interrupt by activating the internal INT signal. This leads to generating an external interrupt signal on the $\overline{\text{INT}}$ pin. The CPU responds with an interrupt acknowledge (INTA) sequence. The leading edge of the first $\overline{\text{INTA}}$ pulse sets an internal interrupt acknowledge signal (we will call it Internal INTA). Internal INTA is reset by the high going edge of the third $\overline{\text{INTA}}$ pulse. The MPSC will not accept any internal requests for an interrupt during the period when Internal $\overline{\text{INTA}}$ is active (high). The MPSC resolves the priority during various existing internal interrupt requests during the Interrupt Request Priority Resolve Time, which is defined as the time between the leading edge of the first INTA and the leading edge of the second INTA from the CPU. Once the internal priorities have been resolved, an internal Interrupt-in-service Latch is set. The external $\overline{\text{INT}}$ is also deactivated when the Interrupt-in-Service Latch is set.

The lower priority interrupt requests are not accepted internally until an EOI (WR0: D5 D4 D3 = 111 Ch. A only) command is issued by the CPU. The EOI command enables the lower priority interrupts. However, a higher priority interrupt request will still be accepted (except during the period when internal $\overline{\text{INTA}}$ is active) even though the Internal-in-Service Latch is set.

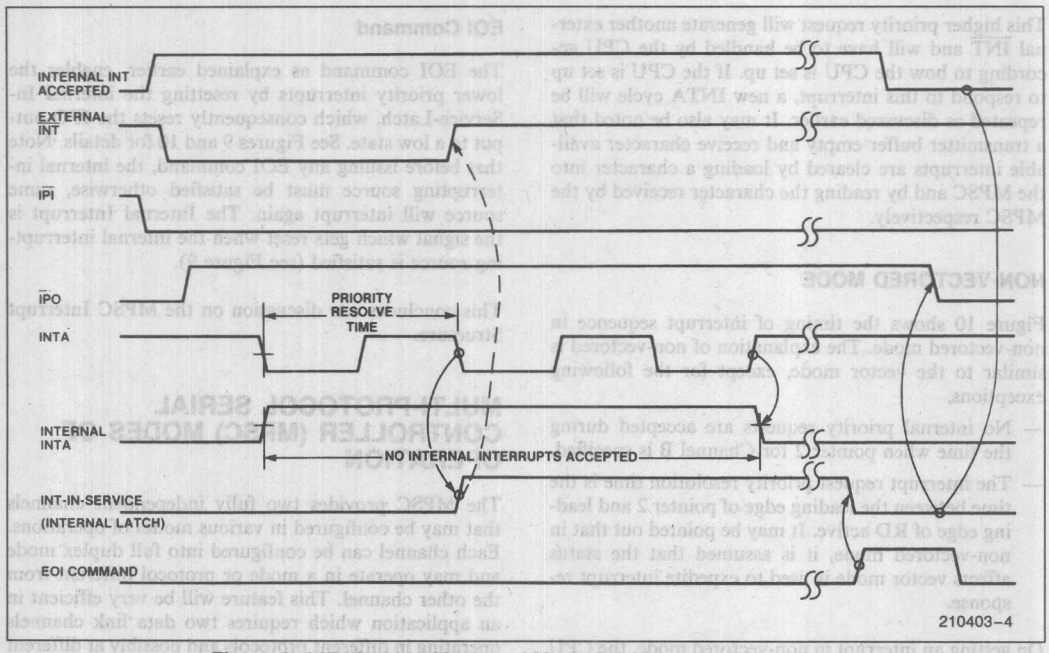


Figure 9. 8274 in 8085 Vectored Mode Priority Resolution Time

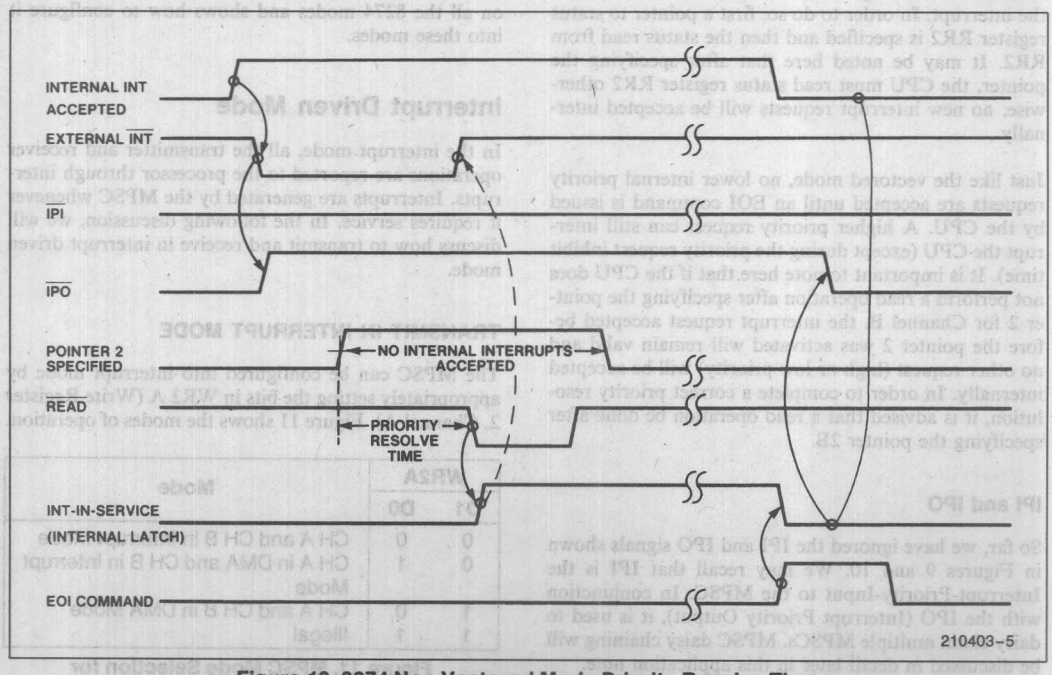


Figure 10. 8274 Non Vectored Mode Priority Resolve Time

This higher priority request will generate another external $\overline{INT}$ and will have to be handled by the CPU according to how the CPU is set up. If the CPU is set up to respond to this interrupt, a new INTA cycle will be repeated as discussed earlier. It may also be noted that a transmitter buffer empty and receive character available interrupts are cleared by loading a character into the MPSC and by reading the character received by the MPSC respectively.

NON-VECTORED MODE

Figure 10 shows the timing of interrupt sequence in non-vectored mode. The explanation of non-vectored is similar to the vector mode, except for the following exceptions.

— No internal priority requests are accepted during the time when pointer 2 for Channel B is specified.

— The interrupt request priority resolution time is the time between the leading edge of pointer 2 and leading edge of RD active. It may be pointed out that in non-vectored mode, it is assumed that the status affects vector mode is used to expedite interrupt response.

On getting an interrupt in non-vectored mode, the CPU must read status register RR2 to find out the cause of the interrupt. In order to do so, first a pointer to status register RR2 is specified and then the status read from RR2. It may be noted here that after specifying the pointer, the CPU must read status register RR2 otherwise, no new interrupt requests will be accepted internally.

Just like the vectored mode, no lower internal priority requests are accepted until an EOI command is issued by the CPU. A higher priority request can still interrupt the CPU (except during the priority request inhibit time). It is important to note here that if the CPU does not perform a read operation after specifying the pointer 2 for Channel B, the interrupt request accepted before the pointer 2 was activated will remain valid and no other request (high or low priority) will be accepted internally. In order to complete a correct priority resolution, it is advised that a read operation be done after specifying the pointer 2B.

IPI and IPO

So far, we have ignored the IPI and IPO signals shown in Figures 9 and 10. We may recall that IPI is the Interrupt-Priority-Input to the MPSC. In conjunction with the IPO (Interrupt Priority Output), it is used to daisy chain multiple MPSCs. MPSC daisy chaining will be discussed in detail later in this application note.

EOI Command

The EOI command as explained earlier, enables the lower priority interrupts by resetting the internal In-Service-Latch, which consequently resets the IPO output to a low state. See Figures 9 and 10 for details. Note that before issuing any EOI command, the internal interrupting source must be satisfied otherwise, same source will interrupt again. The Internal Interrupt is the signal which gets reset when the internal interrupting source is satisfied (see Figure 9).

This concludes our discussion on the MPSC Interrupt Structure.

MULTI-PROTOCOL SERIAL CONTROLLER (MPSC) MODES OF OPERATION

The MPSC provides two fully independent channels that may be configured in various modes of operations. Each channel can be configured into full duplex mode and may operate in a mode or protocol different from the other channel. This feature will be very efficient in an application which requires two data link channels operating in different protocols and possibly at different data rates. This section presents a detailed discussion on all the 8274 modes and shows how to configure it into these modes.

Interrupt Driven Mode

In the interrupt mode, all the transmitter and receiver operations are reported to the processor through interrupts. Interrupts are generated by the MPSC whenever it requires service. In the following discussion, we will discuss how to transmit and receive in interrupt driven mode.

TRANSMIT IN INTERRUPT MODE

The MPSC can be configured into interrupt mode by appropriately setting the bits in WR2 A (Write Register 2, Channel A). Figure 11 shows the modes of operation.

WR2A		Mode
D1	D0	
0	0	CH A and CH B in Interrupt Mode
0	1	CH A in DMA and CH B in Interrupt Mode
1	0	CH A and CH B in DMA Mode
1	1	Illegal

Figure 11. MPSC Mode Selection for Channel A and Channel B

We will limit our discussion to SDLC transmit and receive only. However, exceptions for other synchronous protocols will be pointed out. To initiate a frame transmission, the first data character must be loaded from the CPU, in all cases. (DMA Mode too, as you will notice later in this application note). Note that in SDLC mode, this first data character may be the address of the station addressed by the MPSC. The transmit buffer consists of a transmit buffer and a serial shift register. When the character is transferred from the buffer into the serial shift regiser, an interrupt due to transmit buffer empty is generated. The CPU has one byte time to service this interrupt and load another character into the transmitter buffer. The MPSC will generate an interrupt due to transmit buffer underrun condition if the CPU does not service the Transmit Buffer Empty Interrupt within one byte time.

This process will continue until the CPU is out of any more data characters to be sent. At this point, the CPU does not respond to the interrupt with a character but simply issues a Reset Tx INT/DMA pending command (WR0: D5 D4 D3 = 101). The MPSC will ultimately underrun, which simply means that both the transmit buffer and transmit shift registers are empty. At this point, flag character (7EH) or CRC byte is loaded into the transmit shift register. This sets the transmit underrun bit in RR0 and generates "Transmit Underrun/EOM" interrupt (RR0: D6 = 1).

You will recall that an SDLC frame has two CRC bytes after the data field. 8274 generates the CRC on all the data that is loaded from the CPU. During initialization, there is a choice of selecting a CRC-16 or CCITT-CRC (WR5: D2). In SDLC/HDLC operation, CCITT-CRC must be selected. We will now see how the CRC gets inserted at the end of the data field. Here we have a choice of having the CRC attached to the data field or sending the frame without the CRC bytes. During transmission, a "Reset Tx Underrun/EOM Latch" command (WR0: D7 D6 = 11) will ensure that at the end of the frame when the transmitter underruns, CRC bytes will be automatically inserted at the end of the data field. If the "Reset Tx Underrun/EOM Latch" command was not issued during the transmission of data characters, no CRC would be inserted and the MPSC will transmit flags (7EH) instead.

However, in case of CRC transmission, the CRC transmission sets the Tx Underrun/EOM bit and generates a Transmitter Underrun/EOM Interrupt as discussed earlier. This will have to be reset in the next frame to ensure CRC insertion in the next frame. It is recommended that Tx Underrun/EOM latch be reset very early in the transmission mode, preferably after loading the first character. It may be noted here that Tx Underrun EOM latch cannot be reset if there is no data in the transmit buffer. This means that at least one character has to be loaded into the MPSC before a "Reset Transmit Underrun/EOM Latch" command will be accepted by the MPSC.

When the transmitter is underrun, an interrupt is generated. This interrupt is generated at the beginning of the CRC transmission, thus giving the user enough time (minimum 22 transmit clock cycles) to issue an Abort command (WR0: D5 D4 D3 = 0 0 1) in case if the transmitted data had an error. The Abort Command will ensure that the MPSC transmits at least eight 1's but less than fourteen 1's before the line reverts to continuous flags. The receiver will scratch this frame because of bad CRC.

However, assuming the transmission was good (no Abort Command issued), after the CRC bytes have been transmitted, closing flag (7EH) is loaded into the transmit buffer. When the flag (7EH) byte is transferred to the serial shift register, a transmit buffer empty interrupt is generated. If another frame has to be transmitted, a new data character has to be loaded into the transmit buffer and the complete transmit sequence repeated. If no more frames are to be transmitted, a "Reset Transmit INT/DMA Pending" command (WR0: D5 D4 D3 = 101) will reset the transmit buffer empty interrupt.

For character oriented protocols (Bisync, Monosync), the same discussion is valid, except that during transmit underrun condition and transmit underrun/EOM bit in set state, instead of flags, filler sync characters are transmitted.

CRC Generation

The transmit CRC enable bit (WR5: D0) must be set before loading any data into the MPSC. The CRC generator must be reset to all 1's at the beginning of each frame before CRC computation has begun. The CRC computation starts on the first data character loaded from the CPU and continues until the last data character. The CRC generated is inverted before it is sent on the Tx Data line.

Transmit Termination

A successful transmission can be terminated by issuing a "Reset Transmit Interrupt/DMA Pending" command, as discussed earlier. However, the transmitter may be disabled any time during the transmission and the results will be as shown in Figure 12.

RECEIVE IN INTERRUPT MODE

The receiver has to be initialized into the appropriate receive mode (see sample program later in this application note). The receiver must be programmed into Hunt Mode (WR3: D4) before it is enabled (WR3: D0). The receiver will remain in the Hunt Mode until a flag (or sync character) is received. While in the SDLC/Bisync/Monosync mode, the receiver does not enter the Hunt Mode unless the Hunt bit (WR3, D4) is set again or the receiver is enabled again.

SDLC Address byte is stored in WR6. A global address (FFH) has been hardwired on the MPSC. In address search mode (WR3: D2 = 1), any frame with address matching with the address in WR6 will be received by the MPSC. Frames with global address (FFH) will also be received, irrespective of the condition of address search mode bit (WR3: D2). In general receive mode (WR3: D2 = 0), all frames will be received.

Transmitter Disabled during	Result
1. Data Transmission	Tx Data will send idle characters* which will be zero inserted.
2. CRC Transmission	16 bit transmission, corresponding to 16 bits of CRC will be completed. However, flag bits will be substituted in the CRC field.
3. Immediately after issuing ABORT command.	Abort will still be transmitted—output will be in the mark state.

Figure 12. Transmitter Disabled During Transmission

*NOTE:
Idle characters are defined as a string of 15 or more contiguous ones.

Since the MPSC only recognizes single byte address field, extended address recognition will have to be done by the CPU on the data passed on by the MPSC. If the first address byte is checked by the MPSC, and the CPU determines that the second address byte does not have the correct address field, it must set the Hunt Mode (WR3: D2 = 1) and the MPSC will start searching for a new address byte preceded by a flag.

Programmable Interrupts

The receiver may be programmed into any one of the four modes. See Figure 13 for details.

WR1, CHA		Rx Interrupt Mode
D4	D3	
0	0	Rx INT/DMA disable
0	1	Rx INT on first character
1	0	INT on all Rx characters (Parity affects vector)
1	1	INT on all Rx characters (Parity does not affect vector)

Figure 13. Receiver Interrupt Modes

All receiver interrupts can be disabled by WR1: D4 D3 = 00. Receiver interrupt on first character is normally used to start a DMA transfer or a block transfer sequence using WAIT to synchronize the data transfer to received or transmitted data.

External Status Interrupts

Any change in $\overline{CD}$ input or Abort detection in the received data, will generate an interrupt if External Status Interrupt was enabled (WR1: D0).

Special Receive Conditions

The receiver buffer is quadruply buffered. If the CPU fails to respond to "receive character" available interrupt within a period of three byte times (received bytes), the receiver buffer will overflow and generate an interrupt. Finally, at the end of the received frame, an interrupt will be generated when a valid ending flag has been detected.

Receive Character Length

The receive character length (6, 7 or 8 bits/character) may be changed during reception. However, to ensure that the change is effective on the next received character, this must be done fast enough such that the bits specified for the next character have not been assembled.

CRC Checking

The opening flag in the frame resets the receive CRC generator and any field between the opening and closing flag is checked for the CRC. In case of a CRC error, the CRC/Framing Error bit in status register 1 is set (RR1: D6 = 1). Receiver CRC may be disabled/enabled by WR3,D3. The CRC bytes on the received frame are passed on to the CPU just like data, and may be discarded by the CPU.

Receive Terminator

An end of frame is indicated by End of Frame interrupt. The CPU may issue an "Error Reset" command to reset this interrupt.

DMA (Direct Memory Access) Mode

The 8274 can be interfaced directly to the Intel DMA Controllers 8237A, 8257A and Intel I/O Processor 8089. The 8274 can be programmed into DMA mode by setting appropriate bits in WR2A. See Figure 11 for details.

TRANSMIT IN DMA MODE

After initializing the 8274 into the DMA mode, the first character must be loaded from the CPU to start the DMA cycle. When the first data character (may be the address byte in SDLC) is transferred from the transmit buffer to the transmit serial shift register, the transmit buffer gets empty and a transmit DMA request (TxDRQ) is generated for the channel. Just like the interrupt mode, to ensure that the CRC bytes are included in the frame, the transmit under-run/EOM latch must be reset. This should preferably be done after loading the first character from the CPU. The DMA will progress without any CPU intervention. When the DMA controller reaches the terminal count, it will not respond to the DMA request, thus letting the MPSC under-run. This will ensure CRC transmission. However, the under-run condition will generate an interrupt due to the Tx under-run/EOM bit getting set (RR0: D6). The CPU should issue a "Reset TxInt/DRQ pending" command to reset TxDRQ and issue a "Reset External Status" command to reset Tx Under-run/EOM interrupt. Following the CRC transmission, flag (7EH) will be loaded into the transmit buffer. This will also generate the TxDRQ since the transmit buffer is empty following the transmission of the CRC bytes. The CPU may issue a "Reset TxINT/DRQ pending" command to reset the TxDRQ. "Reset TxINT/DRQ pending" command must be issued before setting up the transmit DMA channel on the DMA Controller, otherwise the MPSC will start the DMA transfer immediately after the DMA channel is set up.

RECEIVE IN DMA MODE

The receiver must be programmed in RxINT on first receive character mode (WR1: D4 D3 = 0 1). Upon receiving the first character, which may be the address byte in SDLC, the MPSC generates an interrupt and also generates a Rx DMA Request (Rx DRQ) for the appropriate channel. The CPU has three byte times to service this interrupt (enable the DMA controller, etc.) before the receiver buffer will overflow. It is advisable to initialize the DMA controller before receiving the first character. In case of high bit rates, the CPU will have to service the interrupt very fast in order to avoid receiver over-run.

Once the DMA is enabled, the received data is transferred to the memory under DMA control. Any received error conditions or external status change condition will generate an interrupt as in the interrupt driven mode. The End of Frame is indicated by the End of Frame interrupt which is generated on reception of the closing flag of the SDLC frame. This End of Frame condition also disables the Receive DMA request. The End of Frame interrupt may be reset by issuing an "Error Reset" command to the MPSC. The "Error Reset" command also re-enables the Receive DMA request. It may be noted that the End of Frame condition sets bit D7 in RR1. This bit gets reset by "Error Reset" command. However, End of Frame bit (RR1: D7) can also be reset by the flag of the next incoming frame. For proper operation, Error Reset Command should be issued "after" the End of Frame Bit (RR1: D7) is set. In a more general case, "Error Reset" command should be issued after End of Frame, Receive over-run or Receive parity bit are set in RR1.

Wait Mode

The wait mode is normally used for block transfer by synchronizing the data transfer through the Ready output from the MPSC, which may be connected to the Ready input of the CPU. The mode can be programmed by WR 1, D7 D5 and may be programmed separately and independently on CH A and CH B. The Wait Mode will be operative if the following conditions are satisfied.

(i) Interrupts are enabled.

(ii) Wait Mode is enabled (WR1: D7)

(iii) CS = 0, A1 = 0

The RDY output becomes active when the transmitter buffer is full or receiver buffer is empty. This way the RDY output from the MPSC can be used to extend the CPU read and write cycle by inserting WAIT states. RDY_A or RDY_B are in high impedance state when the corresponding channel is not selected. This makes it possible to connect RDY_A and RDY_B outputs in wired OR configuration. Caution must be exercised here in using the RDY outputs of the MPSC or else the CPU may hang up for indefinite period. For example, let us assume that transmitter buffer is full and RDY_A is active, forcing the CPU into a wait state. If the $\overline{CTS}$ goes inactive during this period, the RDY_A will remain active for indefinite period and CPU will continue to insert wait states.

Vectored/Non-Vectored Mode

The MPSC is capable of providing an interrupt vector in response to the interrupt acknowledge sequence from the CPU. WR2, CH B contains this vector and the vector can be read in status register RR2. WR2, CH A (bit D5) can program the MPSC in vectored or non-vectored mode. See Figure 14 for details.

In both cases, WR2 may still have the vector stored in it. However, in vectored mode, the MPSC will put the vector on the data bus in response to the INTA (Interrupt Acknowledge) sequence as shown in Figure 15. In non-vectored mode, the MPSC will not respond to the INTA sequence. However, the CPU can read the vector by polling Status Register RR2. WR2A, D4 and D3 can be programmed to respond to 8085 or 8086 INTA sequence. It may be noted here that $\overline{IPI}$ (Interrupt Priority In) pin on the MPSC must be active for the vector to appear on the data bus.

WR2A, D5	Interrupt Mode
0	Non-vectored Interrupt
1	Vectored Interrupt

Figure 14. Vectored Interrupt

WR2A D5	D4	D3	IPI	Mode	1st INTA	2nd INTA	3rd INTA
0	X	X	X	Non-Vectored	HI-Z	HI-Z	HI-Z
1	0	0	0	8085-1	1100 1101	V7 V6 V5 V4 V3 V2 V1 V0	0000 0000
1	0	0	1	8085-1	1100 1101	HI-Z	HI-Z
1	0	1	0	8085-2	HI-Z	V7 V6 V5 V4 V3 V2 V1 V0	0000 0000
1	0	1	1	8085-2	HI-Z	HI-Z	HI-Z
1	1	0	0	8086	HI-Z	V7 V6 V5 V4 V3 V2 V1 V0	—
1	1	0	1	8086	HI-Z	HI-Z	—

Figure 15. MPSC Vectored Interrupts

STATUS AFFECT VECTOR

The Vector stored in WR2B can be modified by the source of the interrupt. This can be done by setting the Status Affect Vector bit (WR1: D2). This powerful feature of the MPSC provides fast interrupt response time, by eliminating the need of writing a routine to read the status of the MPSC. Three bits of the vector are modified in eight different ways as shown on Figure 16. Bits V4, V3, V2 are modified in 8085 based system and bits V2, V1, V0 are modified in 8086/88 based system.

In non-vectored mode, the status affect vector mode can still be used and the vector read by the CPU. Status register RR2B (Read Register 2 in Channel B) will contain this modified vector.

(8085 (8086)	V4 V2	V3 V1	V2 V0	Channel	Interrupt Source
	0	0	0	B	Tx Buffer Empty
	0	0	1		EXT/STAT Change
	0	1	0		RX CHAR Available
	0	1	1		Special Rx Condition
	1	0	0	A	Tx Buffer Empty
	1	0	1		EXT/STAT Change
	1	1	0		RX CHAR Available
	1	1	1		Special Rx Condition

Rx Special Condition: Parity Error, Framing Error, Rx Over-run Error, EOF (SDLC).
EXT/STAT Change: Change in Modem Control Pin Status: CTS, DCD, SYNC, EOM, Break/Abort Detection.

Figure 16. Status Affect Vector Mode

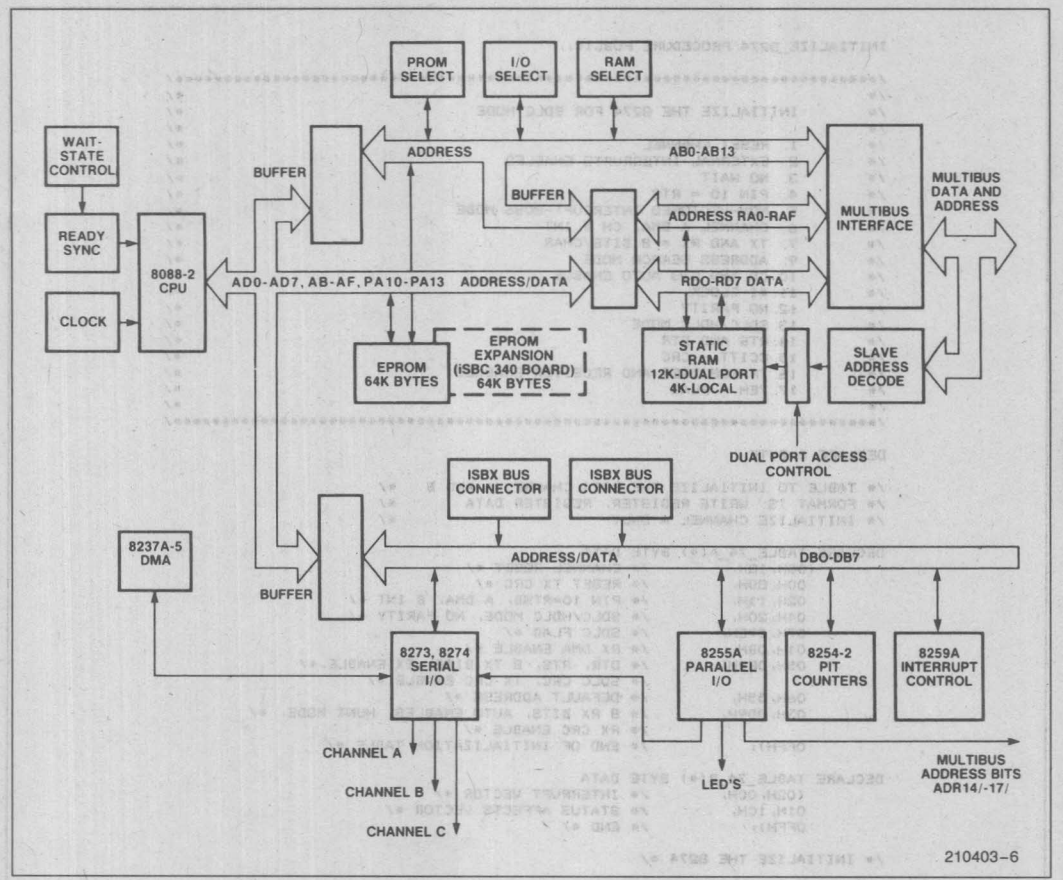


Figure 17. Functional Block Diagram—iSBC® 88/45

APPLICATION EXAMPLE

This section describes the hardware and software of an 8274/8088 system. The hardware vehicle used is the INTEL Single Board Computer iSBC 88/45—Advanced Communication Controller. The software which exercises the 8274 is written in PLM 86. This example will demonstrate how 8274 can be configured into the SDLC mode and transfer data through DMA control. The hardware example will help the reader configure his hardware and the software examples will help in developing an application software. Most software examples closely approximate real data link controller software in the SDLC communication and may be used with very little modification.

iSBC® 88/45

A brief description of the iSBC 88/45 board will be presented here. For more detailed information on the board and the schematics, refer to Hardware Manual for the iSBC 88/45, Advanced Communication Controller. iSBC 88/45 is an intelligent slave/multimaster communication board based on the 8088 processor, the 8274 and the 8273 SDLC/HDLC controller. Figure 17 shows the functional block diagram of the board. The iSBC 88/45 has the following features.

- 8 MHz processor
- 16K bytes of static RAM (12K dual port)
- Multimaster/Intelligent Slave Multibus Interface
- Nine Interrupt Levels 8259A
- Two serial channels through 8274
- One Serial channel through 8273
- S/W programmable baud rate generator
- Interfaces: RS232, RS422/449, CCITT V.24
- 8237A DMA controller
- Baud Rate to 800K Baud

```
INITIALIZE_8274: PROCEDURE PUBLIC;

/*************************************************************************/
/*                                                                       */
/*       INITIALIZE THE 8274 FOR SDLC MODE                               */
/*                                                                       */
/*       1.  RESET CHANNEL                                               */
/*       2.  EXTERNAL INTERRUPTS ENABLED                                 */
/*       3.  NO WAIT                                                     */
/*       4.  PIN 10 = RTS                                                */
/*       5.  NON-VECTORED INTERRUPT-8086 MODE                            */
/*       6.  CHANNEL A DMA, CH B INT                                     */
/*       7.  TX AND RX = 8 BITS/CHAR                                     */
/*       9.  ADDRESS SEARCH MODE                                         */
/*       10. CD AND CTS AUTO ENABLE                                      */
/*       11. X1 CLOCK                                                    */
/*       12. NO PARITY                                                   */
/*       13. SDLC/HDLC MODE                                              */
/*       14. RTS AND DTR                                                 */
/*       15. CCITT - CRC                                                 */
/*       16. TRANSMITTER AND RECEIVER ENABLED                            */
/*       17. 7EH = FLAG                                                  */
/*                                                                       */
/*************************************************************************/

DECLARE C BYTE;

/* TABLE TO INITIALIZE THE 8274 CHANNEL A AND B    */
/* FORMAT IS:  WRITE REGISTER, REGISTER DATA        */
/* INITIALIZE CHANNEL A ONLY                        */

DECLARE TABLE_74_A(*) BYTE DATA
       (00H, 18H,          /* CHANNEL RESET */
        00H, 80H,          /* RESET TX CRC */
        02H, 11H,          /* PIN 10=RTSB, A DMA, B INT */
        04H, 20H,          /* SDLC/HDLC MODE, NO PARITY */
        07H, 07EH,         /* SDLC FLAG */
        01H, 08H,          /* RX DMA ENABLE */
        05H, 0EBH,         /* DTR, RTS, 8 TX BITS, TX ENABLE, */
                           /* SDLC CRC, TX CRC ENABLE */
        06H, 55H,          /* DEFAULT ADDRESS */
        03H, 0D9H,         /* 8 RX BITS, AUTO ENABLES, HUNT MODE, */
                           /* RX CRC ENABLE */
        0FFH);             /* END OF INITIALIZATION TABLE */

DECLARE TABLE_74_B(*) BYTE DATA
        (02H, 00H,         /* INTERRUPT VECTOR */
        01H, 1CH,          /* STATUS AFFECTS VECTOR */
        0FFH);             /* END */

/* INITIALIZE THE 8274 */

C=0;
DO WHILE TABLE_74_B(C) <> 0FFH;
        OUTPUT(COMMAND_B_74) = TABLE_74_B(C);
        C=C+1;
        OUTPUT(COMMAND_B_74) = TABLE_74_B(C);
        C=C+1;
END;

C=0;
DO WHILE TABLE_74_A(C) <> 0FFH;
        OUTPUT(COMMAND_A_74) = TABLE_74_A(C);
        C=C+1;
        OUTPUT(COMMAND_A_74) = TABLE_74_A(C);
        C=C+1;
END;
RETURN;
END INITIALIZE_8274;
```

210403-7

Figure 18. Typical MPSC SDLC Initialization Sequence

For this application, the CPU is run at 8 MHz. The board is configured to operate the 8274 in SDLC operation with the data transfer in DMA mode using the 8237A. 8274 is configured first in non-vectored mode in which case the INTEL Priority Interrupt Controller 8259A is used to resolve priority between various interrupting sources on the board and subsequently interrupt the CPU. However, the vectored mode of the 8274 is also verified by disabling the 8259A and reading the vectors from the 8274. Software examples for each case will be shown later.

The application example is interrupt driven and uses DMA for all data transfers under 8237A control. The 8254 provides the transmit and receive clocks for the 8274. The 8274 was run at 400K baud with a local loopback (jumper wire) on Channel A data. The board was also run at 800K baud by modifying the software as will be discussed later in the Special Applications section. One detail to note is that the Rx Channel DMA request line from the 8274 has higher priority than the Tx Channel DMA request line. The 8274 master clock was 4.0 MHz. The on-board RAM is used to define transmit and receive data buffers. In this application, the data is read from memory location 800H through 810H and transferred to memory location 900H to 910H through the 8274 Serial Link. The operation is full duplex. 8274 modem control pins, $\overline{CTS}$ and $\overline{CD}$ have been tied low (active).

Software

The software consists of a monitor program and a program to exercise the 8274 in the SDLC mode. Appendix A contains the entire program listing. For the sake of clarity, each source module has been rewritten in a simple language and will be discussed here individually. Note that some labels in the actual listings in the Appendix will not match with the labels here. Also the listing in the Appendix sets up some flags to communicate with the monitor. Some of these flags are not explained in detail for the reason that they are not pertinent to this discussion. The monitor takes the command from a keyboard and executes this program, logging any error condition which might occur.

8274 Initialization

The MPSC is initialized in the SDLC mode for Channel A. Channel B is disabled. See Figure 18 for the initialization routine. Note that WR4 is initialized before setting up the transmitter and receive parameters. However, it may also be pointed out that other than WR4, all the other registers may be programmed in any order. Also SDLC-CRC has been programmed for correct operation. An incorrect CRC selection will result *in incorrect operation.* Also note that receive interrupt on first receive character has been programmed although Channel A is in the DMA mode.

Interrupt Routines

The 8274 interrupt routines will be discussed here. On an 8274 interrupt, program branches off to the "Main Interrupt Routine". In main interrupt routine, status register RR2 is read. RR2 contains the modified vector. The cause of the interrupt is determined by reading the modified bits of the vector. Note that the 8274 has been programmed in the non-vectored mode and status affects vector bit has been set. Depending on the value of the modified bits, the appropriate interrupt routine is called. See Figure 19 for the flow diagram and Figure 20 for the source code. Note that an End of Interrupt Command is issued after servicing the interrupt. This is necessary to enable the lower priority interrupts.

Figure 21 shows all the interrupt routines called by the Main Interrupt Routine. "Ignore-Interrupt" as the name implies, ignores any interrupts and sets the FAIL flag. This is done because this program is for Channel A only and we are ignoring any Channel B interrupts. The important thing to note is the Channel A Receiver Character available routine. This routine is called after receiving the first character in the SDLC frame. Since the transfer mode is DMA, we have a maximum of three character times to service this interrupt by enabling the DMA controller.

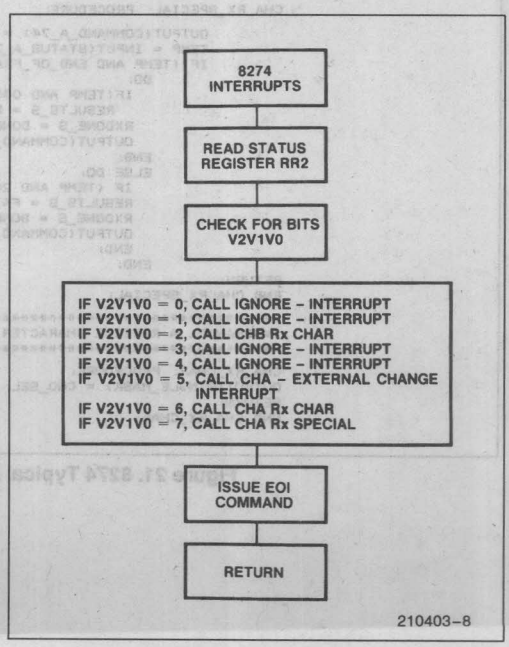


Figure 19. Interrupt Response Flow Diagram

```
/*************************/
/* MAIN INTERRUPT ROUTINE */
/*************************/

OUTPUT(COMMAND_B_74) = 2;                    /* SET POINTER TO 2*/
TEMP = INPUT(STATUS_B_74) AND 07H;           /* READ INTERRUPT VECTOR */
                                             /* CHECK FOR CHA INT ONLY*/
/* FOR THIS APPLICATION CH B INTERRUPTS ARE IGNORED*/
DO CASE TEMP;
        CALL    IGNORE_INT;                  /* V2V1V0 = 000*/
        CALL    IGNORE_INT;                  /* V2V1V0 = 001*/
        CALL    CHB_RX_CHAR;                 /* V2V1V0 = 010*/
        CALL    IGNORE_INT;                  /* V2V1V0 = 011*/
        CALL    IGNORE_INT;                  /* V2V1V0 = 100*/
        CALL    CHA_EXTERNAL_CHANGE;         /* V2V1V0 = 101*/
        CALL    CHA_RX_CHAR;                 /* V2V1V0 = 110*/
        CALL    CHA_RX_SPECIAL;              /* V2V1V0 = 111*/
END;
OUTPUT(COMMAND_A_74) =38H;       /* END OF INTERRUPT FOR 8274 */
RETURN;
END INTERRUPT_8274;
```

210403-9

Figure 20. Typical Main Interrupt Routine

```
/*********************************************************/
/* CHANNEL A EXTERNAL/STATUS CHANGE INTERRUPT HANDLER */
/*********************************************************/
CHA_EXTERNAL_CHANGE: PROCEDURE;

TEMP = INPUT(STATUS_A_74);              /* STATUS REG 1*/
IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
        TXDONE_S=DONE;
ELSE DO;
        TXDONE_S=DONE;
        RESULTS_S=FAIL;
    END;
OUTPUT(COMMAND_A_74) = 10H;      /* RESET EXT/STATUS INTERRUPTS */
RETURN;
END CHA_EXTERNAL_CHANGE;
/*********************************************************/
/* CHANNEL A SPECIAL RECEIVE CONDITIONS INTERRUPT HANDLER */
/*********************************************************/

CHA_RX_SPECIAL: PROCEDURE;

        OUTPUT(COMMAND_A_74) = 1;
        TEMP = INPUT(STATUS_A_74);
        IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN
            DO;
              IF(TEMP AND 040H) = 040H THEN
                RESULTS_S = FAIL;            /* CRC ERROR */
              RXDONE_S = DONE;
              OUTPUT(COMMAND_A_74) = 30H; /*ERROR RESET*/
            END;
            ELSE DO;
              IF (TEMP AND 20H) = 20H THEN DO;
              RESULTS_S = FAIL;              /* RX OVERRUN ERROR*/
              RXDONE_S = DONE;
              OUTPUT(COMMAND_A_74) = 30H; /*ERROR RESET*/
              END;
            END;
RETURN;
END CHA_RX_SPECIAL;
/*************************************/
/* CHANNEL A RECEIVE CHARACTER AVAILABLE */
/*************************************/

CHA_RX_CHAR: PROCEDURE;
OUTPUT(SINGLE_MASK) = CH0_SEL;          /*ENABLE RX DMA CHANNEL*/
RETURN;
END CHA_RX_CHAR;
```

210403-10

Figure 21. 8274 Typical Interrupt Handling Routines

It may be recalled that the receiver buffer is three bytes deep in addition to the receiver shift register. At very high data rates, it may not be possible to have enough time to read RR2, enable the DMA controller without overrunning the receiver. In a case like this, the DMA controller may be left enabled before receiving the Receive Character Interrupt. Remember, the Rx DMA request and interrupt for the receive character appears at the same time. If the DMA controller is enabled, it would service the DMA request by reading the received character. This will make the 8274 interrupt line go inactive. However, the 8259A has latched the interrupt and a regular interrupt acknowledge sequence still occurs after the DMA controller has completed the transfer and given up the bus. The 8259A will return Level 7 interrupt since the 8274 interrupt has gone away. The user software must take this into account, otherwise the CPU will hang up.

The procedure shown for the Special Receive Condition Interrupt checks if the interrupt is due to the End of Frame. If this is not TRUE, the FAIL flag is set and the program aborted. For a real life system, this must be followed up by error recovery procedures which obviously are beyond the scope of this Application Note.

The transmission is terminated when the End of Message (RR0, D6) interrupt is generated. This interrupt is serviced in the Channel A External/Status Change interrupt procedure. For any other change in external status conditions, the program is aborted and a FAIL flag set.

Main Program

Finally, we will briefly discuss the main program. Figure 22 shows the source program. It may be noted that the Transmit Under-run latch is reset after loading the first character into the 8274. This is done to ensure CRC transmission at the end of the frame. Also, the first character is loaded from the CPU to start DMA transfer of subsequent data. This concludes our discussion on hardware and software example. Appendix A also includes the software written to exercise the 8274 in the vectored mode by disabling the 8259A.


```
CHA_SDLC_TEST: PROCEDURE BYTE PUBLIC;

      CALL    ENABLE_INTERRUPTS_S;
      CALL    INIT_8274_SDLC_S;
      ENABLE;
      OUTPUT(COMMAND_A_74) = 28H;  /* RESET TX INT/DMA          */
      OUTPUT(COMMAND_B_74) = 28H;  /* BEFORE INITIALIZING 8237*/
      CALL    INIT_8237_S;
      OUTPUT(DATA_A_74) = 55H;     /*LOAD FIRST CHARACTER FROM */
                                   /*CPU */
    /* TO ENSURE CRC TRANSMISSION, RESET TX UNDERRUN LATCH   */
      OUTPUT(COMMAND_A_74) = 0C0H;
      RXDONE_S, TXDONE_S=NOT_DONE;  /* CLEAR ALL FLAGS        */
      RESULTS_S=PASS;               /* FLAG SET FOR MONITOR   */
      DO WHILE TXDONE_S=NOT_DONE;   /* DO UNTIL TERMINAL COUNT */
      END;

      DO WHILE(INPUT(STATUS_A_74) AND 04H) <> 04H;
      /* WAIT FOR CRC TO GET TRANSMITTED */
      /* TEST FOR TX BUFFFER EMPTY TO VERIFY THIS*/
      END;
      DO WHILE RXDONE_S=NOT_DONE;  /* DO UNTIL TERMINAL COUNT */
      END;
      CALL    STOP_8237_S;
END CHA_SDLC_TEST;
```

210403-11

Figure 22. Typical 8274 Transmit/Receive Set-Up in SDLC Mode

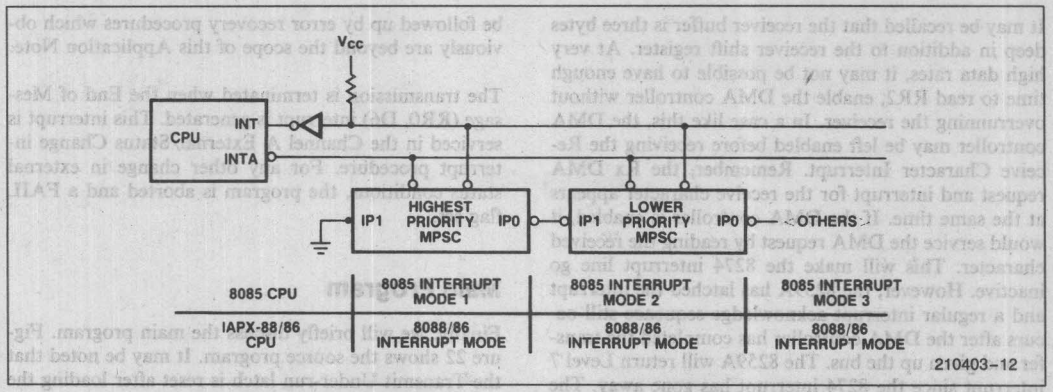


Figure 23. 8274 Daisy Chain Vectored Mode

SPECIAL APPLICATIONS

In this section, some special application issues will be discussed. This will be useful to a user who may be using a mode which is possible with the 8274 but not explicitly explained in the data sheet.

MPSC Daisy Chain Operation

Multiple MPSCs can be connected in a daisy-chain configuration (see Figure 23). This feature may be useful in an application where multiple communication channels may be required and because of high data rates, conventional interrupt controller is not used to avoid long interrupt response times. To configure the MPSCs for the daisy chain operation, the interrupt priority input pins ($\overline{IPI}$) and interrupt priority output pins ($\overline{IPO}$) of the MPSC should be connected as shown. The highest priority device has its $\overline{IPI}$ pin connected to ground. Each MPSC is programmed in a vectored mode with status affects vector bit set. In the 8085 basic systems, only one MPSC should be programmed in the 8085 Mode 1. This is the MPSC which will put the call vector (CD Hex) on the data bus in response to the first $\overline{INTA}$ pulse (see Figure 15). It may be pointed out that the MPSC in 8085 Mode 1 will provide the call vector irrespective of the state of $\overline{IPI}$ pin. Once a higher priority MPSC generates an interrupt, its $\overline{IPO}$ pin goes inactive thus preventing lower priority MPSCs from interrupting the CPU. Preferably the highest priority MPSC should be programmed in 8085 Mode 1. It may be recalled that the Priority Resolve Time on a given MPSC extends from the falling edge of the first $\overline{INTA}$ pulse to the falling edge of the second $\overline{INTA}$ pulse. During this period, no new internal interrupt requests are accepted. The maximum number of the MPSCs that can be connected in a daisy chain is limited by the Priority Resolution Time. Figure 24 shows a maximum number of MPSCs that can be connected in various CPU systems.

It may be pointed out that $\overline{IOP}$ to $\overline{IPI}$ delay time specification is 100 ns.

System Configuration	Priority Resolution Time Min (ns)	Number of 8274s Daisy Chained (Max)
8086-1	400	4
8086-2	500	5
8086	800	8
8088	800	8
8085-2	1200	12
8085A	1920	19

NOTE:
Zero wait states have been assumed.

Figure 24. 8274 Daisy Chain Operation

Bisync Transparent Communication

Bisync applications generally require that data transparency be established during communication. This requires that the special control characters may not be included in the CRC accumulation. Refer to the Synchronous Protocol Overview section for a more detailed discussion on data transparency. The 8274 can be used for transparent communication in Bisync communications. This is made possible by the capability of the MPSC to selectively turnon/turnoff the CRC accumulation while transmitting or receiving. In bisync transparent transmit mode, the special characters (DLE, DLE SYN, etc) are excluded from CRC calculation. This can be easily accomplished by turning off the transmit CRC calculation (WR5: D5 = 0) before loading the special character into the transmit buffer. If the next character is to be included in the CRC accumulation, then the CRC can be enabled (WR5: D5 = 1). See Figure 25 for a typical flow diagram.

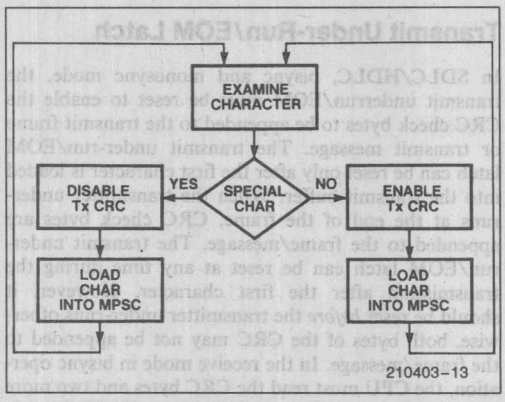


Figure 25. Transmit in Bisync Transparent Mode

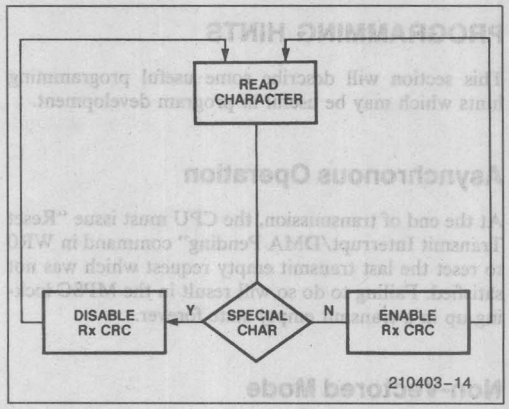


Figure 26. Receive in Bisync Transparent Mode

During reception, it is possible to exclude received character from CRC calculation by turning off the Receive CRC after reading the special character. This is made possible by the fact that the received data is presented to receive CRC checker 8 bit times after the character has been received. During this 8 bit times, the CPU must read the character and decide if it wants to be included in the CRC calculation. Figure 26 shows the typical flow diagram to achieve this.

It should be noted that the CRC generator must be enabled during CRC reception. Also, after reading the CRC bytes, two more characters (SYNC) must be read before checking for CRC check result in RR1.

Auto Enable Mode

In some data communication applications, it may be required to enable the transmitter or the receiver when the $\overline{CTS}$ or the $\overline{CD}$ lines respectively, are activated by the modems. This may be done very easily by programming the 8274 into the Auto Enable Mode. The auto enable mode is set by writing a '1' to WR3,D5. The function of this mode is to enable the transmitter automatically when $\overline{CTS}$ goes active. The receiver is enabled when $\overline{CD}$ goes active. An in-active state of $\overline{CTS}$ or $\overline{CD}$ pin will disable the transmitter or the receiver respectively. However, the Transmit Enable bit (WR5:D3) and Receive Enable bit (WR3:D1) must be set in order to use the auto enable mode. In non-auto mode, the transmitter or receiver is enabled if the corresponding bits are set in WR5 and WR3, irrespective of the state $\overline{CTS}$ or $\overline{CD}$ pins. It may be recalled that any transition on $\overline{CTS}$ or $\overline{CD}$ pin will generate External/Status Interrupt with the corresponding bits set in RR1. This interrupt can be cleared by issuing a Reset External/Status interrupt command as discussed earlier.

Note that in auto enable mode, the character to be transmitted must be loaded into the transmit buffer after the $\overline{CTS}$ becomes active, not before. Any character loaded into the transmit buffer before the $\overline{CTS}$ became active will not be transmitted.

High Speed DMA Operation

In the section titled Application Example, the MPSC has been programmed to operate in DMA mode and receiver is programmed to generate an interrupt on the first receive character. You may recall that the receive FIFO is three bytes deep. On receiving the interrupt on the first receive character, the CPU must enable the DMA controller within three received byte times to avoid receiver over-run condition. In the application example, at 400K baud, the CPU had approximately 60 μs to enable the DMA controller to avoid receiver buffer overflow. However, at higher baud rates, the CPU may not have enough time to enable the DMA controller in time. For example, at 1M baud, the CPU should enable the DMA controller within approximately 24 μs to avoid receiver buffer overrun. In most applications, this is not sufficient time. To solve this problem, the DMA controller should be left enabled before getting the interrupt on the first receive character (which is accompanied by the Rx DMA request for the appropriate channel). This will allow he DMA controller to start DMA transfer as soon as the Rx DMA request becomes active without giving the CPU enough time to respond to the interrupt on the first receive character. The CPU will respond to the interrupt *after* the DMA transfer has been completed and will find the 8259A (see Application Example) responding with interrupt level 7, the lowest priority level. Note that the 8274 interrupt request was satisfied by the DMA controller, hence the interrupt on the first receive character was cleared and the 8259A had no pending interrupt. Because of no pending interrupt, the 8259A returned interrupt level 7 in response to the INTA sequence from the CPU. The user software should take care of this interrupt.

PROGRAMMING HINTS

This section will describe some useful programming hints which may be useful in program development.

Asynchronous Operation

At the end of transmission, the CPU must issue "Reset Transmit Interrupt/DMA Pending" command in WR0 to reset the last transmit empty request which was not satisfied. Failing to do so will result in the MPSC locking up in a transmit empty state forever.

Non-Vectored Mode

In non-vectored mode, the Interrupt Acknowledge pin (INTA) on the MPSC must be tied high through a pull-up resistor. Failing to do so will result in unpredictable response from the 8274.

HDLC/SDLC Mode

When receiving data in SDLC mode, the CRC bytes must be read by the CPU (or DMA controller) just like any other data field. Failing to do so will result in receiver buffer overflow. Also, the End of Frame Interrupt indicates that the entire frame has been received. At this point, the CRC result (RR1:D6) and residue code (RR1:D3, D2, D1) may be checked.

Status Register RR2

ChB RR2 contains the vector which gets modified to indicate the source of interrupt (see the section titled MPSC Modes of Operation). However, the state of the vector does not change if no new interrupts are generated. The contents of ChB RR2 are only changed when a new interrupt is generated. In order to get the correct information, RR2 must be read only after an interrupt is generated, otherwise it will indicate the previous state.

Initialization Sequence

The MPSC initialization routine must issue a channel Reset Command at the beginning. WR4 should be defined before other registers. At the end of the initialization sequence, Reset External/Status and Error Reset commands should be issued to clear any spurious interrupts which may have been caused at power up.

Transmit Under-Run/EOM Latch

In SDLC/HDLC, bisync and monosync mode, the transmit underrun/EOM must be reset to enable the CRC check bytes to be appended to the transmit frame or transmit message. The transmit under-run/EOM latch can be reset only after the first character is loaded into the transmit buffer. When the transmitter underruns at the end of the frame, CRC check bytes are appended to the frame/message. The transmit underrun/EOM latch can be reset at any time during the transmission after the first character. However, it should be reset *before* the transmitter under-runs otherwise, both bytes of the CRC may not be appended to the frame/message. In the receive mode in bisync operation, the CPU must read the CRC bytes and two more SYNC characters before checking for valid CRC result in RR1.

Sync Character Load Inhibit

In bisync/monosync mode only, it is possible to prevent loading sync characters into the receive buffers by setting the sync character load inhibit bit (WR3:D1 = 1). Caution must be exercised in using this option. It may be possible to get a CRC character in the received message which may match the sync character and not get transferred to the receive buffer. However, sync character load inhibit should be enabled during all pre-frame sync characters so the software routine does not have to read them from the MPSC.

In SDLC/HDLC mode, sync character load inhibit bit must be reset to zero for proper operation.

EOI Command

EOI command can only be issued through channel A irrespective of which channel had generated the interrupt.

Priority in DMA Mode

There is no priority in DMA mode between the following four singals: TxDRQ(CHA), RxDRQ(CHA), TxDRQ(CHB), RxDRQ(CHB). The priority between these four signals must be resolved by the DMA controller. At any given time, all four DMA channels from the 8274 are capable of going active.

APPENDIX A
APPLICATION EXAMPLE: SOFTWARE LISTINGS

```
PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST


SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE INIT_8274_S
OBJECT MODULE PLACED IN :F1:SINI74.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:SINI74.PLM TITLE(iSBC 88/45 8274 CHANNEL
A SDLC TEST) COMPACT NOINTVECTOR ROM


                /*************************************************/
                /*                                               */
                /*       INITIALIZE THE 8274 FOR SDLC MODE       */
                /*                                               */
                /*       1.  RESET CHANNEL                       */
                /*       2.  EXTERNAL INTERRUPTS ENABLED         */
                /*       3.  NO WAIT                             */
                /*       4.  PIN 10 = RTS                        */
                /*       5.  NON-VECTORED INTERRUPT-8086 MODE    */
                /*       6.  CHANNEL A DMA,  CH B INT            */
                /*       7.  TX AND RX = 8 BITS/CHAR             */
                /*       9.  ADDRESS SEARCH MODE                 */
                /*       10. CD AND CTS AUTO ENABLE              */
                /*       11. X1 CLOCK                            */
                /*       12. NO PARITY                           */
                /*       13. SDLC/HDLC MODE                      */
                /*       14. RTS AND DTR                         */
                /*       15. CCITT - CRC                         */
                /*       16. TRANSMITTER AND RECEIVER ENABLED    */
                /*       17. 7EH = FLAG                          */
                /*                                               */
                /*************************************************/

  1             INIT_8274_S:  DO;

                $INCLUDE (:F1:PORTS.PLM)

              = /*************************************************/
              = /*                                               */
              = /*       iSBC 88/45 PORT ASSIGNMENTS             */
              = /*                                               */
              = /*************************************************/

  2   1       = DECLARE LIT LITERALLY 'LITERALLY';

              = /* 8237A-5  PORTS */

  3   1       = DECLARE CH0_ADDR          LIT     '080H',
              =         CH0_COUNT         LIT     '081H',
              =         CH1_ADDR          LIT     '082H',
              =         CH1_COUNT         LIT     '083H',
              =         CH2_ADDR          LIT     '084H',
              =         CH2_COUNT         LIT     '085H',
              =         CH3_ADDR          LIT     '086H',
              =         CH3_COUNT         LIT     '087H',
              =         STATUS_37         LIT     '088H',
              =         COMMAND_37        LIT     '088H',
              =         REQUEST_REG_37    LIT     '089H',
              =         SINGLE_MASK       LIT     '08AH',
              =         MODE_REG_37       LIT     '08BH',

PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

              =         CLR_BYTE_PTR_37   LIT     '08CH',
              =         TEMP_REG_37       LIT     '08DH',
              =         MASTER_CLEAR_37   LIT     '08DH',
              =         ALL_MASK_37       LIT     '08FH';

              = /* 8254-2 PORTS */

  4   1       = DECLARE CTR_00            LIT     '090H',
              =         CTR_01            LIT     '091H',
              =         CTR_02            LIT     '092H',
```

210403-15

```
                =           CONTROL0_54     LIT     '093H',
                =           STATUS0_54      LIT     '093H',
                =           CTR_10          LIT     '098H',
                =           CTR_11          LIT     '099H',
                =           CTR12           LIT     '09AH',
                =           CONTROL1_54     LIT     '09BH',
                =           STATUS1_54      LIT     '09BH';

                =   /* 8255 PORTS */

      5   1     =   DECLARE PORTA_55        LIT     '0A0H',
                =           PORTB_55        LIT     '0A1H',
                =           PORTC_55        LIT     '0A2H',
                =           CONTROL_55      LIT     '0A3H';

                =   /* 8274 PORTS */

      6   1     =   DECLARE DATA_A_74       LIT     '0D0H',
                =           DATA_B_74       LIT     '0D1H',
                =           STATUS_A_74     LIT     '0D2H',
                =           COMMAND_A_74    LIT     '0D2H',
                =           STATUS_B_74     LIT     '0D3H',
                =           COMMAND_B_74    LIT     '0D3H';

                =   /* 8259A PORTS */

      7   1     =   DECLARE STATUS_POLL_59  LIT     '0E0H',
                =           ICW1_59         LIT     '0E0H',
                =           OCW2_59         LIT     '0E0H',
                =           OCW3_59         LIT     '0E0H',
                =           OCW1_59         LIT     '0E1H',
                =           ICW2_59         LIT     '0E1H',
                =           ICW3_59         LIT     '0E1H',
                =           ICW4_59         LIT     '0E1H';

                =   /* 8274 REGISTER BIT ASSIGNMENTS */
                =   /* READ REGISTER 0 */

      8   1     =   DECLARE RX_AVAIL        LIT     '01H',
                =           INT_PENDING     LIT     '02H',
                =           TX_EMPTY        LIT     '04H',
                =           CARRIER_DETECT  LIT     '08H',
                =           SYNC_HUNT       LIT     '10H',
                =           CLEAR_TO_SEND   LIT     '20H',

PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

                =           END_OF_TX_MESSAGE LIT   '40H',
                =           BREAK_ABORT     LIT     '80H';

                =   /* READ REGISTER 1 */

      9   1     =   DECLARE ALL_SENT        LIT     '01H',
                =           PARITY_ERROR    LIT     '10H',
                =           RX_OVERRUN      LIT     '20H',
                =           CRC_ERROR       LIT     '40H',
                =           END_OF_FRAME    LIT     '80H';

                =   /* READ REGISTER 2 */

     10   1     =   DECLARE TX_B_EMPTY      LIT     '00H',
                =           EXT_B_CHANGE    LIT     '01H',
                =           RX_B_AVAIL      LIT     '02H',
                =           RX_B_SPECIAL    LIT     '03H',
                =           TX_A_EMPTY      LIT     '04H',
                =           EXT_A_CHANGE    LIT     '05H',
                =           RX_A_AVAIL      LIT     '06H',
                =           RX_A_SPECIAL    LIT     '07H';
```

210403-16

```
                =   /* 8237 BIT ASSIGNMENTS */

     11    1    =   DECLARE CH0_SEL          LIT     '00H',
                =           CH1_SEL          LIT     '01H',
                =           CH2_SEL          LIT     '02H',
                =           CH3_SEL          LIT     '03H',
                =           WRITE_XFER       LIT     '04H',
                =           READ_XFER        LIT     '08H',
                =           DEMAND_MODE      LIT     '00H',
                =           SINGLE_MODE      LIT     '40H',
                =           BLOCK_MODE       LIT     '80H',
                =           SET_MASK         LIT     '04H';

     12    1        DELAY_S: PROCEDURE PUBLIC;
     13    2        DECLARE D WORD;
     14    2        D=0;
     15    2        DO WHILE D<800H;
     16    3        D=D+1;
     17    3        END;
     18    2        END DELAY_S;


     19    1        INIT_8274_SDLC_S:   PROCEDURE PUBLIC;

     20    2        DECLARE C   BYTE;

                    $EJECT


PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST


                    /* TABLE TO INITIALIZE THE 8274 CHANNEL A AND B */

                    /*  FORMAT IS: WRITE REGISTER, REGISTER DATA  */
                    /*       INITIALIZE CHANNEL ONLY                */

     21    2        DECLARE TABLE_74_A(*) BYTE DATA
                            (00H,18H,       /* CHANNEL RESET */
                            00H,80H,        /* RESET TX CRC */
                            02H,11H,        /* PIN 10=RTSB, A DMA, B INT */
                            04H,20H,        /* SDLC/HDLC MODE, NO PARITY */
                            07H,07EH,       /* SDLC FLAG */
                            01H,0BH,        /* RX DMA ENABLE */
                            05H,0EBH,       /* DTR, RTS, 8 TX BITS, TX ENABLE, TX CRC ENABLE */
                            06H,55H,        /* DEFAULT ADDRESS */
                            03H,0D9H,       /* 8 RX BITS, AUTO ENABLES, HUNT MODE, */
                                            /* RX CRC ENABLE */
                            0FFH);          /* END OF INITIALIZATION TABLE */

     22    2        DECLARE TABLE_74_B(*) BYTE DATA
                            (02H,00H,       /* INTERRUPT VECTOR */
                            01H,1CH,        /* STATUS AFFECTS VECTOR */
                            0FFH);          /* END */


                    /* INITIALIZE THE 8254 */

     23    2        OUTPUT(CONTROLO_54)=36H;
     24    2        OUTPUT(CTR_00) = LOW(20);       /* BAUD RATE = 400K_BAUD*/
     25    2        OUTPUT(CTR_00) = HIGH(20);      /* BAUD RATE = 400K_BAUD*/


                    /* INITIALIZE THE 8274 */

     26    2        C=0;
     27    2        DO WHILE TABLE_74_B(C) <> 0FFH;
     28    3            OUTPUT(COMMAND_B_74) = TABLE_74_B(C);
     29    3            C=C+1;
     30    3            OUTPUT(COMMAND_B_74) = TABLE_74_B(C);
     31    3            C=C+1;
     32    3        END;
```

210403-17

```
  33   2      C=0;
  34   2      DO WHILE TABLE_74_A(C) <> 0FFH;
  35   3          OUTPUT(COMMAND_A_74) = TABLE_74_A(C);
  36   3          C=C+1;
  37   3          OUTPUT(COMMAND_A_74) = TABLE_74_A(C);
  38   3          C=C+1;
  39   3      END;
  40   2          CALL    DELAY_S;

  41   2      RETURN;
  42   2      END INIT_8274_SDLC_S;
  43   1      END INIT_8274_S;



PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

MODULE INFORMATION:

     CODE AREA SIZE     = 00A8H    168D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0003H      3D
     MAXIMUM STACK SIZE = 0006H      6D
     213 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION



PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE INIT_8237_CHA
OBJECT MODULE PLACED IN :F1:SINI37.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:SINI37.PLM TITLE(iSBC 88/45 8274 CHANNEL A SDLC
TEST) COMPACT NOINTVECTOR ROM

                /*************************************************************************/
                /*                                                                       */
                /*      8237    INITIALIZATION ROUTINE  FOR DMA TRANSFER                 */
                /*                                                                       */
                /*************************************************************************/

   1            INIT_8237_CHA: DO;

                $NOLIST

  12   1       INIT_8237_S: PROCEDURE PUBLIC;

  13   2       OUTPUT(MASTER_CLEAR_37)=0;
  14   2       OUTPUT(COMMAND_37) = 20H;          /* EXTENDED WRITE */
  15   2       OUTPUT(ALL_MASK_37) = 0FH;         /* MASK ALL REQUESTS */
  16   2       OUTPUT(MODE_REG_37) = (SINGLE_MODE OR WRITE_XFER OR CH0_SEL);
  17   2       OUTPUT(MODE_REG_37) = (SINGLE_MODE OR READ_XFER OR CH1_SEL);
  18   2       OUTPUT(CLR_BYTE_PTR_37) = 0;
  19   2       OUTPUT(CH0_ADDR) = 00;        /* RECEIVE BUFF AT 900H */
  20   2       OUTPUT(CH0_ADDR) = 09H;
  21   2       OUTPUT(CH0_COUNT) = 0H;
  22   2       OUTPUT(CH0_COUNT) = 01;
  23   2       OUTPUT(CH1_ADDR) = 00;        /* TRANSMIT BUFF AT 800H */
  24   2       OUTPUT(CH1_ADDR) = 08H;
  25   2       OUTPUT(CH1_COUNT) = 010H;
  26   2       OUTPUT(CH1_COUNT) = 00H;
```

210403-18

```
                /* ENABLE TRANSFER */
 27   2         OUTPUT(SINGLE_MASK) = CH1_SEL;    /* ENABLE TX DMA */
 28   2         RETURN;

 29   2         END INIT_8237_S;

                /* TURN OFF THE 8237 CHANNELS 0 AND 1 */

 30   1         STOP_8237_S: PROCEDURE PUBLIC;
 31   2         OUTPUT(SINGLE_MASK) = CH1_SEL OR SET_MASK;
 32   2         OUTPUT(SINGLE_MASK) = CH0_SEL OR SET_MASK;
 33   2         RETURN;
 34   2         END STOP_8237_S;
 35   1         END INIT_8237_CHA;


MODULE INFORMATION:

     CODE AREA SIZE     = 004CH     76D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0000H      0D


PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST


     MAXIMUM STACK SIZE = 0002H      2D
     163 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION


PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST


SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE INTR_8274_S
OBJECT MODULE PLACED IN :F1:SINTR.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:SINTR.PLM TITLE(iSBC 88/45 8274 CHANNEL
A SDLC TEST) COMPACT NOINTVECTOR ROM

                /*****************************************/
                /*                                       */
                /*      8274 INTERRUPT ROUTINE           */
                /*                                       */
                /*****************************************/

   1            INTR_8274_S: DO;
                $NOLIST
  12   1         DECLARE TEMP BYTE;
  13   1         DECLARE (RESULTS_S,TXDONE_S,RXDONE_S) BYTE EXTERNAL;
  14   1         DECLARE INT_VEC POINTER AT (140);
  15   1         DECLARE INT_VEC_STORE POINTER;
  16   1         DECLARE MASK_59 BYTE;
  17   1         DECLARE DONE              LIT     'OFFH',
                       NOT_DONE          LIT     '00H',
                       PASS              LIT     'OFFH',
                       FAIL              LIT     '00H';

                /*****************************/
                /* IGNORE INTERRUPT HANDLER */
                /*****************************/

  18   1         IGNORE_INT: PROCEDURE;

  19   2         RESULTS_S = FAIL;
  20   2         RETURN;
  21   2         END IGNORE_INT;
```

210403-19

```
                /*******************************************************/
                /* CHANNEL A EXTERNAL/STATUS CHANGE INTERRUPT HANDLER */
                /*******************************************************/

   22   1       CHA_EXTERNAL_CHANGE: PROCEDURE;

   23   2       TEMP = INPUT(STATUS_A_74);          /* STATUS REG 1*/
   24   2       IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
   25   2           TXDONE_S=DONE;
   26   2       ELSE DO;
   27   3           TXDONE_S=DONE;
   28   3           RESULTS_S=FAIL;
   29   3            END;
   30   2       OUTPUT(COMMAND_A_74) = 10H; /* RESET EXT/STATUS INTERRUPTS */
   31   2       RETURN;
   32   2       END CHA_EXTERNAL_CHANGE;

                $EJECT

PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

                /*******************************************************/
                /* CHANNEL A SPECIAL RECEIVE CONDITIONS INTERRUPT HANDLER */
                /*******************************************************/

   33   1       CHA_RX_SPECIAL: PROCEDURE;

   34   2           OUTPUT(COMMAND_A_74) = 1;
   35   2           TEMP = INPUT(STATUS_A_74);
   36   2           IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN
   37   2                       DO;
   38   3                         IF(TEMP AND 040H) = 040H THEN
   39   3                           RESULTS_S = FAIL;          /* CRC ERROR */
   40   3                         RXDONE_S = DONE;
   41   3                         OUTPUT(COMMAND_A_74) = 30H; /*ERROR RESET*/
   42   3                       END;
   43   2                       ELSE DO;
   44   3                         IF (TEMP AND 20H) = 20H THEN DO;
   46   4                         RESULTS_S = FAIL;          /* RX OVERRUN ERROR*/
   47   4                         RXDONE_S = DONE;
   48   4                         OUTPUT(COMMAND_A_74) = 30H; /*ERROR RESET*/
   49   4                         END;
   50   3                       END;
   51   2       RETURN;
   52   2       END CHA_RX_SPECIAL;


                /*****************************************/
                /* CHANNEL A RECEIVE CHARACTER AVAILABLE */
                /*****************************************/

   53   1       CHA_RX_CHAR: PROCEDURE;
   54   2       OUTPUT(SINGLE_MASK) = CH0_SEL;          /*ENABLE RX DMA CHANNEL*/
   55   2       RETURN;
   56   2       END CHA_RX_CHAR;

                $EJECT

PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST

                /* ENABLE 8274 INTERRUPTS - SET UP THE 8259A */

   57   1       ENABLE_INTERRUPTS_S: PROCEDURE PUBLIC;

   58   2       DECLARE CHA_INT_ON LIT '0F7H';

   59   2       DISABLE;

   60   2       CALL    SET$INTERRUPT(39, INT_39);
```

210403-20

```
 61   2      INT_VEC_STORE = INT_VEC;
 62   2      INT_VEC = INTERRUPT$PTR(INT_8274_S);
 63   2      MASK_59 = INPUT(OCW1_59);

 64   2      OUTPUT(OCW1_59) = MASK_59 AND CHA_INT_ON;

 65   2      RETURN;
 66   2      END ENABLE_INTERRUPTS_S;


              /* DISABLE 8274 INTERRUPTS - SET UP THE 8259A */

 67   1      DISABLE_INTERRUPTS_S: PROCEDURE PUBLIC;

 68   2      DISABLE;

 69   2      INT_VEC = INT_VEC_STORE;

 70   2      OUTPUT(OCW1_59) = MASK_59;
 71   2      ENABLE;

 72   2      RETURN;
 73   2      END DISABLE_INTERRUPTS_S;

              /* CHANNEL B RECEIVE CHARACTER AVAILABLE */

 74   1      CHB_RX_CHAR: PROCEDURE;

 75   2      TEMP=INPUT(DATA_B_74);

 76   2      OUTPUT(COMMAND_B_74) = 38H;
 77   2      RETURN;
 78   2      END CHB_RX_CHAR;

              $EJECT
PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST


              /**************************/
              /* MAIN INTERRUPT ROUTINE */
              /**************************/

 79   1      INT_8274_S: PROCEDURE INTERRUPT 35 PUBLIC;

 80   2      OUTPUT(COMMAND_B_74) = 2;                  /* SET POINTER TO 2*/
 81   2      TEMP = INPUT(STATUS_B_74) AND 07H;         /* READ INTERRUPT VECTOR */
                                                        /* CHECK FOR CHA INT ONLY*/
              /* FOR THIS APPLICATION CH B INTERRUPTS ARE IGNORED*/
 82   2      DO CASE TEMP;
 83   3          CALL    IGNORE_INT;                /* V2V1V0 = 000*/
 84   3          CALL    IGNORE_INT;                /* V2V1V0 = 001*/
 85   3          CALL    CHB_RX_CHAR;               /* V2V1V0 = 010*/
 86   3          CALL    IGNORE_INT;                /* V2V1V0 = 011*/
 87   3          CALL    IGNORE_INT;                /* V2V1V0 = 100*/
 88   3          CALL    CHA_EXTERNAL_CHANGE;       /* V2V1V0 = 101*/
 89   3          CALL    CHA_RX_CHAR;               /* V2V1V0 = 110*/
 90   3          CALL    CHA_RX_SPECIAL;            /* V2V1V0 = 111*/
 91   3      END;
 92   2      OUTPUT(COMMAND_A_74) =38H;   /* END OF INTERRUPT FOR 8274 */
 93   2      OUTPUT(OCW2_59) = 63H;       /* 8259 EOI */
 94   2      OUTPUT(OCW1_59) = INPUT(OCW1_59) AND 0F7H;
 95   2      RETURN;
 96   2      END INT_8274_S;



              /* DEFAULT INTERRUPT ROUTINE - 8259A INTERRUPT 7 */
              /* REQUIRED ONLY WHEN DMA CONTROLLER IS ENABLED  */
              /* BEFORE RECEIVING FIRST CHARACTER WHICH IS     */
              /* AT HIGH BAUD RATES  LIKE 800K BAUD.  READ APP.*/
              /* NOTE SECTION 6 FOR DETAILS                    */
```

210403-21

```
 97   1      INT_39: PROCEDURE INTERRUPT 39;
 98   2              OUTPUT(OCW2_59) = 20H;    /* NON-SPECIFIC EOI */
 99   2              OUTPUT(OCW1_59) = INPUT(OCW1_59) AND 0F7H;
100   2              RESULTS_S = FAIL;
101   2      END INT_39;

102   1      END INTR_8274_S;


MODULE INFORMATION:

     CODE AREA SIZE     = 01BFH    447D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0006H      6D
     MAXIMUM STACK SIZE = 0022H     34D
     295 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION

PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST


SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE STEST
OBJECT MODULE PLACED IN :F1:STEST.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:STEST.PLM TITLE(iSBC 88/45 8274 CHANNEL A SDLC TEST)
COMPACT NOINTVECTOR ROM

              /*************************************************************************/
              /*                                                                       */
              /*      iSBC 545 PORT A (8274) SDLC TEST                                 */
              /*                                                                       */
              /*************************************************************************/

  1           STEST: DO;


  2   1       DELAY_S: PROCEDURE EXTERNAL;
  3   2       END DELAY_S;

  4   1       ENABLE_INTERRUPTS_S: PROCEDURE EXTERNAL;
  5   2       END ENABLE_INTERRUPTS_S;

  6   1       DISABLE_INTERRUPTS_S: PROCEDURE EXTERNAL;
  7   2       END DISABLE_INTERRUPTS_S;

  8   1       INIT_8274_SDLC_S: PROCEDURE EXTERNAL;
  9   2       END INIT_8274_SDLC_S;

 10   1       INIT_8237_S: PROCEDURE EXTERNAL;
 11   2       END INIT_8237_S;

 12   1       STOP_8237_S: PROCEDURE EXTERNAL;
 13   2       END STOP_8237_S;

 14   1       VERIFY_TRANSFER_S: PROCEDURE EXTERNAL;
 15   2       END VERIFY_TRANSFER_S;

 16   1       INT_8274_S: PROCEDURE INTERRUPT 35 EXTERNAL;
 17   2       END INT_8274_S;
              $NOLIST
              $EJECT

PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST


 28   1       DECLARE (RESULTS_S, TXDONE_S, RXDONE_S) BYTE PUBLIC;
 29   1       DECLARE DONE            LIT     'OFFH',
                      NOT_DONE        LIT     '00H',
                      PASS            LIT     'OFFH',
                      FAIL            LIT     '00H';
```

210403-22

```
          $EJECT

PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST


  30   1      CHA_SDLC_TEST: PROCEDURE BYTE PUBLIC;


  31   2              CALL    ENABLE_INTERRUPTS_S;
  32   2              CALL    INIT_8274_SDLC_S;
  33   2              ENABLE;
  34   2              OUTPUT(COMMAND_A_74) = 28H;   /* RESET TX INT/DMA          */
  35   2              OUTPUT(COMMAND_B_74) = 28H;   /* BEFORE INITIALIZING 8237*/
  36   2              CALL    INIT_8237_S;
  37   2              OUTPUT(DATA_A_74) = 55H;   /* LOAD FIRST CHARACTER FROM CPU*/

                      /* TO ENSURE CRC TRANSMISSION RESET TX UNDERRUN LATCH*/
  38   2              OUTPUT(COMMAND_A_74) = 0C0H;
  39   2              RXDONE_S, TXDONE_S=NOT_DONE;        /* CLEAR ALL FLAGS */
  40   2              RESULTS_S=PASS;                     /* FLAG SET FOR MONITOR*/

  41   2              DO WHILE TXDONE_S=NOT_DONE;         /* DO UNTIL TERMINAL COUNT*/
  42   3              END;

  43   2              DO WHILE(INPUT(STATUS_A_74) AND 04H) <> 04H;
                      /* WAIT FOR CRC TO GET TRANSMITTED */
                      /* TEST FOR TX BUFFFER EMPTY TO VERIFY THIS*/
  44   3              END;
  45   2              DO WHILE RXDONE_S=NOT_DONE;         /* DO UNTIL TERMINAL COUNT*/
  46   3              END;

  47   2              CALL    STOP_8237_S;

  48   2              CALL    DISABLE_INTERRUPTS_S;

  49   2              CALL    VERIFY_TRANSFER_S;

  50   2              RETURN RESULTS_S;

  51   2      END CHA_SDLC_TEST;
  52   1      END STEST;


MODULE INFORMATION:

     CODE AREA SIZE     = 0063H     99D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0003H      3D
     MAXIMUM STACK SIZE = 0004H      4D
     198 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION


PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST


SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE VECTOR_MODE
OBJECT MODULE PLACED IN :F1:VECTOR.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:VECTOR.PLM TITLE(iSBC 88/45 8274 CHANNEL A SDLC TEST)


                /*****************************************************************/
                /*                                                               */
                /*          8274 INTERRUPT HANDLING ROUTINE FOR                  */
                /*              8274 VECTOR MODE                                 */
                /*            STATUS AFFECTS VECTOR                              */
                /*                                                               */
                /*****************************************************************/
```

210403-23

```
                /* THIS IS AN EXAMPLE OF HOW 8274 CAN BE USED IN VECTORED MODE      */
                /* THE iSBC88/45 BOARD WAS REWIRED TO DISABLE THE PIT 8259A AND     */
                /* ENABLE THE 8274 TO PLACE ITS VECTOR ON THE DATABUS IN RESPONSE   */
                /* TO THE INTA SEQUENCE FROM THE 8088.  OTHER MODIFICATIONS INCLUDED */
                /* CHANGES TO 8274 INITIALIZATION PROGRAM (SINI74) TO PROGRAM 8274  */
                /* INTO VECTORED MODE (WRITE REGISTER 2A D5=1).                     */

  1             VECTOR_MODE: DO;
                $NOLIST

 12   1       DECLARE TEMP BYTE;
 13   1       DECLARE (RESULTS_S,TXDONE,RXDONE) BYTE EXTERNAL;
 14   1       DECLARE DONE  LITERALLY 'OFFH',
                       NOT_DONE LITERALLY '00H',
                       PASS     LITERALLY 'OFFH',
                       FAIL     LITERALLY '00H';

                /*********************************************************************/
                /*      TRANSMIT INTERRUPT CHANNEL A INTERRUPT WILL NOT BE SEEN IN THE */
                /*      DMA OPERATION.                                               */
                /*********************************************************************/

 15   1           TX_INTERRUPT_CHA: PROCEDURE INTERRUPT 84;
 16   2           OUTPUT(COMMAND_A_74) = 00101000B;        /*RESET TXINT PENDING*/
 17   2           OUTPUT(COMMAND_A_74) = 00111000B;        /*EOI*/
 18   2           END TX_INTERRUPT_CHA;

                /*********************************************************************/
                /*      EXTERNAL/STATUS INTERRUPT PROCEDURE: CHECKS FOR END OF MESSAGE */
                /*      ONLY.  IF THIS IS NOT TRUE THEN THE FAIL FLAG IS SET.  HOWEVER,  */
                /*      A USER PROGRAM SHOULD CHECK FOR OTHER EXT/STATUS CONDITIONS    */
                /*      ALSO IN RR1 AND THEN TAKE APPROPRIATE ACTION BASED ON THE      */
                /*      APPLICATION.                                                 */
                /*********************************************************************/

 19   1       EXT_STAT_CHANGE_CHA: PROCEDURE INTERRUPT 85;
 20   2              TEMP  = INPUT(STATUS_A_74);
 21   2              IF (TEMP AND END_OF_TX_MESSAGE) = END_OF_TX_MESSAGE THEN
 22   2                 TXDONE = DONE;
 23   2              ELSE DO;
 24   3                 TXDONE = DONE;

PL/M-86 COMPILER    iSBC 88/45 8274 CHANNEL A SDLC TEST

 25   3                    RESULTS_S = FAIL;
 26   3                 END;

 27   2           OUTPUT(COMMAND_A_74) = 00010000B;        /*RESET EXT STAT INT*/
 28   2              OUTPUT(COMMAND_A_74) = 00111000B;        /*EOI*/
 29   2              RETURN;
 30   2       END EXT_STAT_CHANGE_CHA;

                /*********************************************************************/
                /*      RECEIVER CHARACTER AVAILABLE INTERRUPT WILL APPEAR ONLY ON FIRST*/
                /*      RECEIVE CHARACTER.  SINCE DMA CONTROLLER HAS BEEN ENABLED BEFORE */
                /*      THE FIRST CHARACTER IS RECEIVED, THE RECEIVER REQUEST IS      */
                /*      SERVICED BY THE DMA CONTROLLER.                              */
                /*********************************************************************/

 31   1       RX_CHAR_AVAILABLE_CHA: PROCEDURE INTERRUPT 86;
 32   2           OUTPUT(COMMAND_A_74) = 00111000B;        /*EOI*/
 33   2              RETURN;
 34   2       END RX_CHAR_AVAILABLE_CHA;
                $EJECT
```

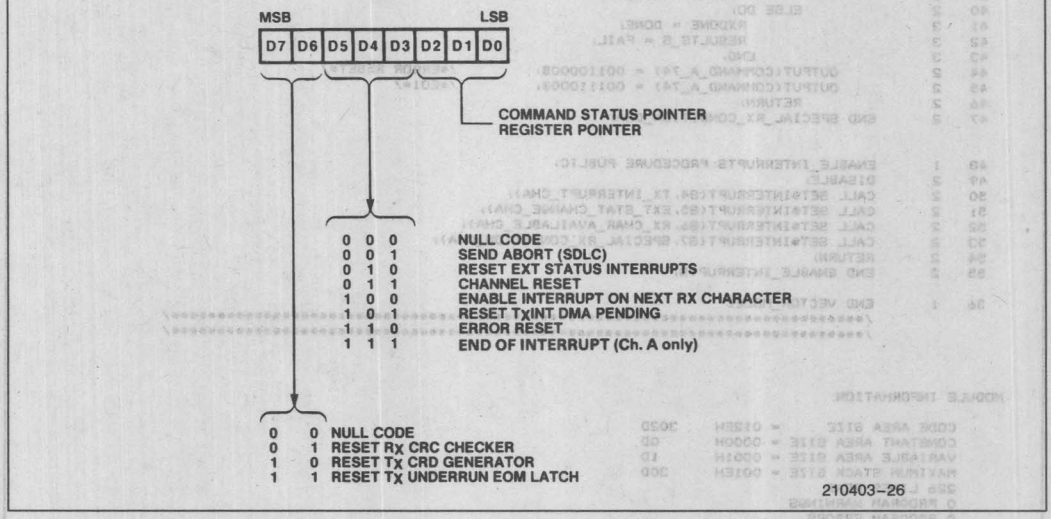
210403-24

```
PL/M-86 COMPILER     iSBC 88/45 8274 CHANNEL A SDLC TEST


                /*************************************************************************/
                /*      SPECIAL RECEIVE CONDITION INTERRUPT SERVICE ROUTINE CHECKS FOR   */
                /*      END OF FRAME BIT ONLY.  SEE SPECIAL SERVICE ROUTINE FOR NON-     */
                /*      VECTORED MODE FOR CRC CHECK AND OVERRUN ERROR CHECK.             */
                /*************************************************************************/

  35   1       SPECIAL_RX_CONDITION_CHA: PROCEDURE INTERRUPT 87;

  36   2             OUTPUT(COMMAND_A_74) = 1;                /*POINTER 1*/
  37   2             TEMP = INPUT(STATUS_A_74);
  38   2             IF (TEMP AND END_OF_FRAME) = END_OF_FRAME THEN
  39   2                   RXDONE = DONE;
  40   2             ELSE DO;
  41   3                   RXDONE = DONE;
  42   3                   RESULTS_S = FAIL;
  43   3                  END;
  44   2          OUTPUT(COMMAND_A_74) = 00110000B;        /*ERROR RESET*/
  45   2          OUTPUT(COMMAND_A_74) = 00111000B;        /*EOI*/
  46   2             RETURN;
  47   2       END SPECIAL_RX_CONDITION_CHA;


  48   1       ENABLE_INTERRUPTS: PROCEDURE PUBLIC;
  49   2       DISABLE;
  50   2       CALL SET$INTERRUPT(84, TX_INTERRUPT_CHA);
  51   2       CALL SET$INTERRUPT(85, EXT_STAT_CHANGE_CHA);
  52   2       CALL SET$INTERRUPT(86, RX_CHAR_AVAILABLE_CHA);
  53   2       CALL SET$INTERRUPT(87, SPECIAL_RX_CONDITION_CHA);
  54   2       RETURN;
  55   2       END ENABLE_INTERRUPTS;

  56   1       END VECTOR_MODE;
                /*************************************************************************/
                /*************************************************************************/


MODULE INFORMATION:

     CODE AREA SIZE     = 012EH    302D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0001H      1D
     MAXIMUM STACK SIZE = 001EH     30D
     226 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

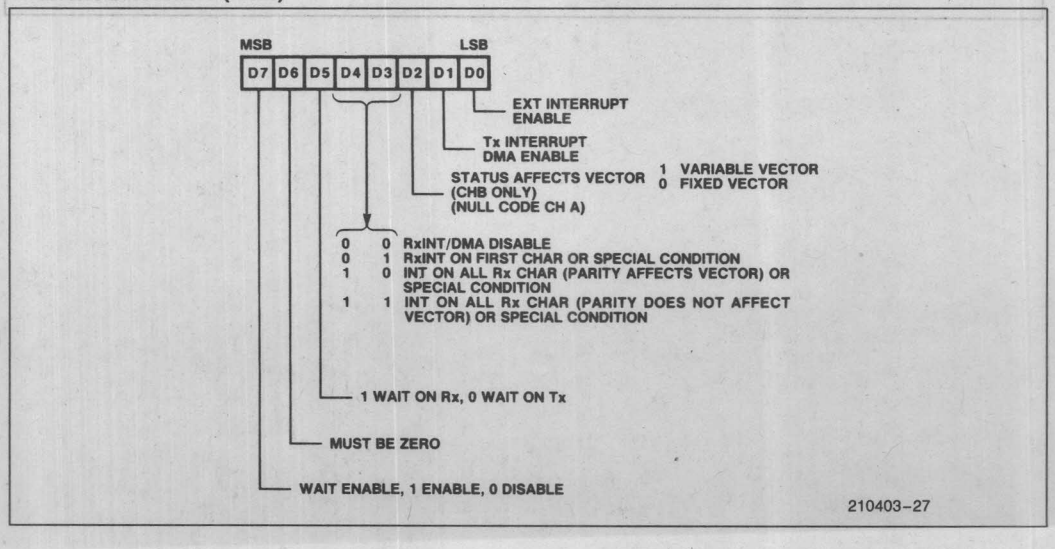
210403-25

APPENDIX B
MPSC READ/WRITE REGISTER DESCRIPTIONS

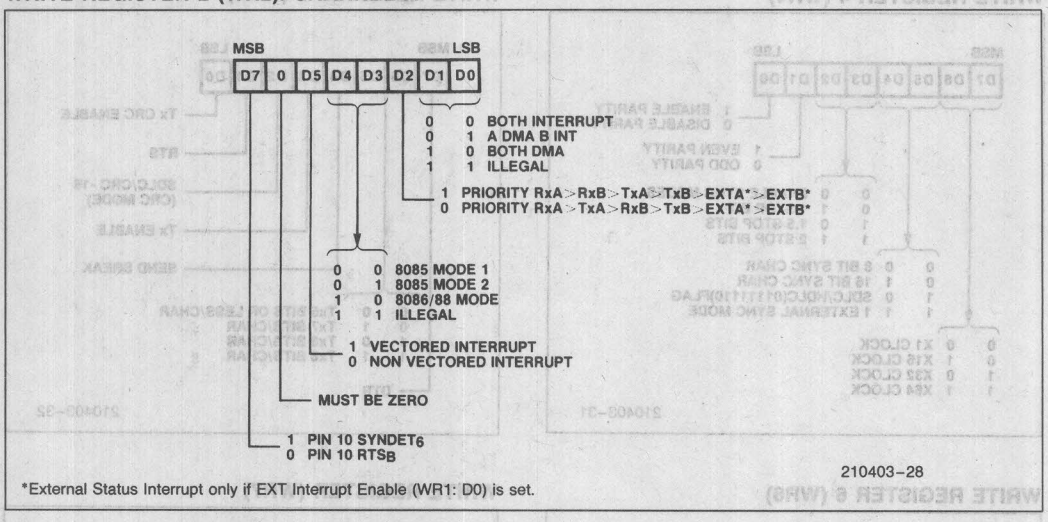
WRITE REGISTER 0 (WR0)



WRITE REGISTER 1 (WR1)

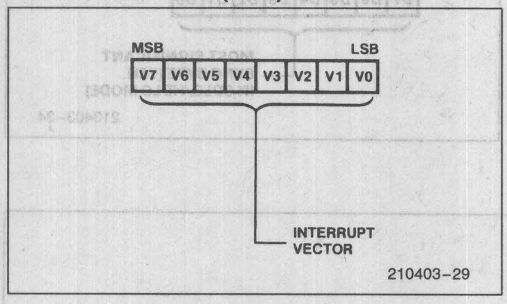


WRITE REGISTER 2 (WR2): CHANNEL A

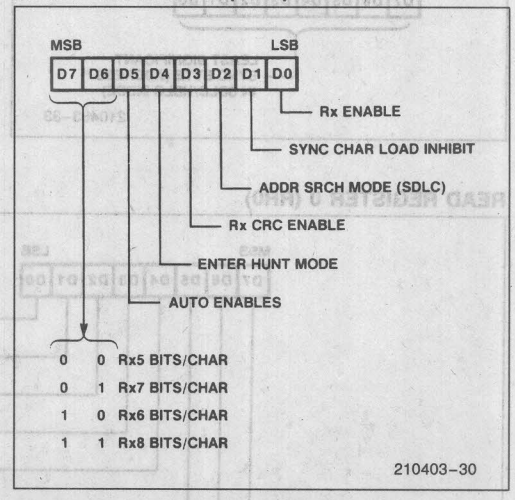


*External Status Interrupt only if EXT Interrupt Enable (WR1: D0) is set.

WRITE REGISTER 2 (WR2): CHANNEL B

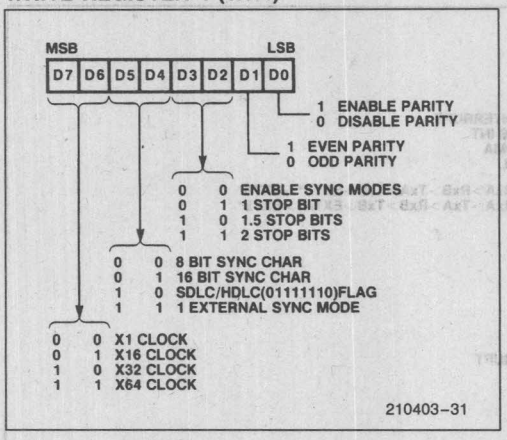


WRITE REGISTER 3 (WR3)

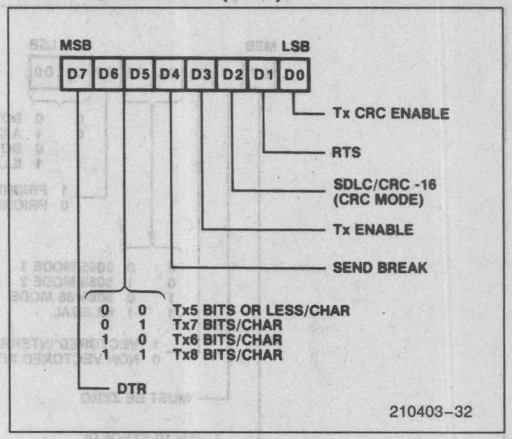


2

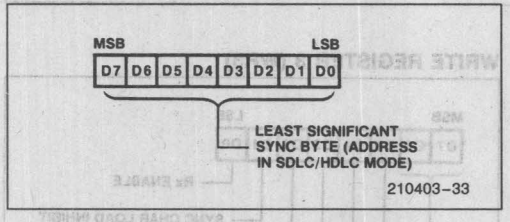
WRITE REGISTER 4 (WR4)



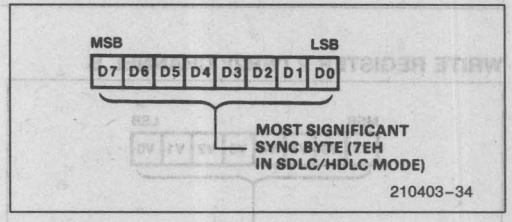
WRITE REGISTER 5 (WR5)



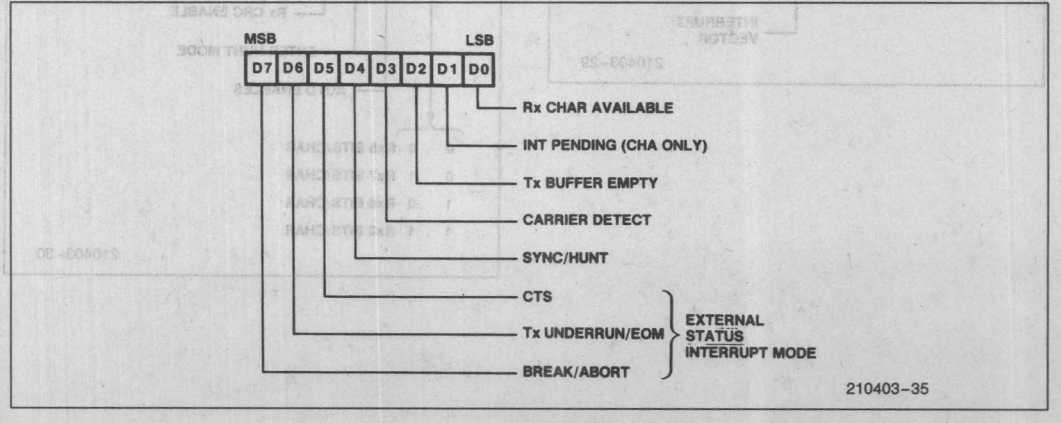
WRITE REGISTER 6 (WR6)



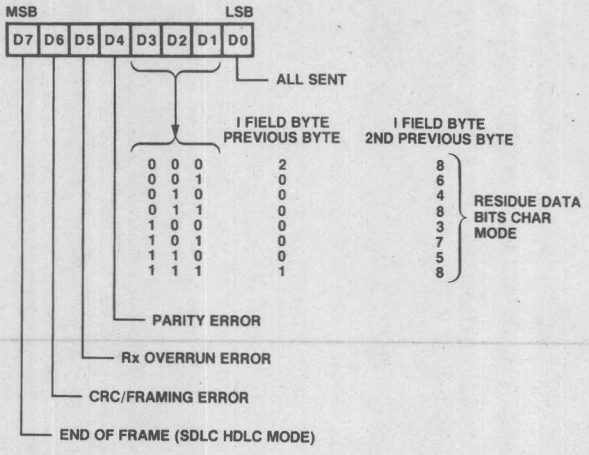
WRITE REGISTER (WR7)



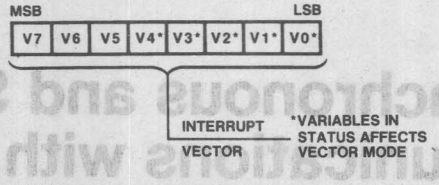
READ REGISTER 0 (RR0)



READ REGISTER 1 (RR1): (SPECIAL RECEIVE CONDITION MODE)



READ REGISTER 2 (RR2) CHANNEL B ONLY



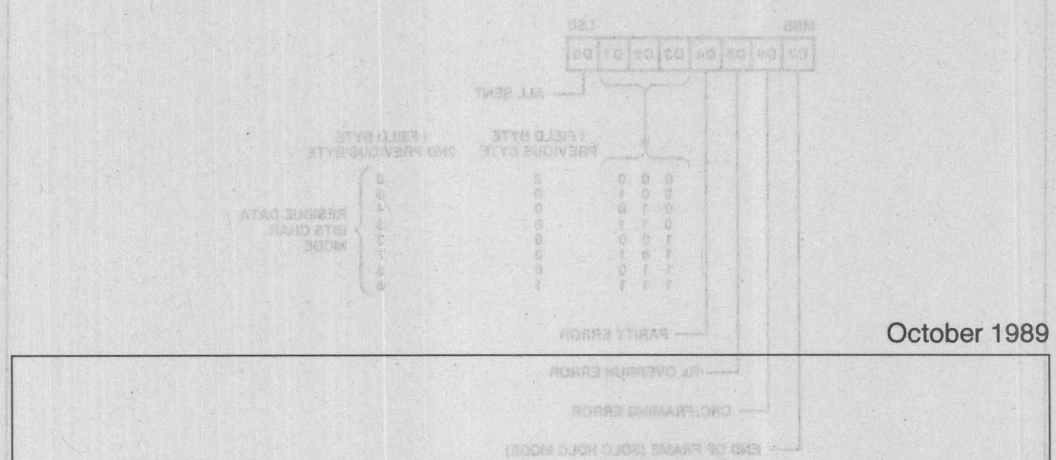
REFERENCES

1. *IBM Document No. GA27-3004-2: General Information—Binary Synchronous Communications*
2. *Application Note AP134: Asynchronous Communication with the 8274 Multiple Protocol Serial Controller.* Intel Corp., Ca.
3. *8274 MPSC Data Sheet,* Intel Corporation, Ca.
4. *iSBC 88/45 Hardware Reference Manual,* Intel Corp., Ca.
5. *Computer Networks and Distributed Processing by James Martin.* Prentice Hall, Inc., N.J.

intel®

APPLICATION NOTE

AP-222



October 1989

Asynchronous and SDLC Communications with 82530

DFG TECHNICAL MARKETING

Order Number: 231262-004

ASYNCHRONOUS AND SDLC COMMUNICATIONS WITH 82530

CONTENTS

PAGE

INTRODUCTION 2-428
1. SCC Port Definition 2-428
2. Accessing the SCC Registers 2-428
3. Initialization fo ASYNC Operation 2-430
4. ASYNC Communication in Polling Mode 2-430
5. ASYNC Communication in Interrupt Mode 2-431
6. Initialization for SDLC Communication 2-433
7. SDLC Frame Reception 2-433
8. SDLC Frame Transmission 2-433
9. SDLC Interrupt Routines 2-436

CONCLUSIONS 2-436

REFERENCES 2-436

APPENDIX A: 82530-BAUD RATE GENERATORS 2-437

APPENDIX B: MODEM CONTROL PINS ON THE 82350 2-439

APPENDIX C: INTERFACING 82530 TO 80186 2-441

INTRODUCTION

INTEL's 82530, Serial Communications Controller (SCC), is a dual channel, multi-protocol data communications peripheral. It is designed to interface to high speed communications lines using asynchronous, byte synchronous, and bit synchronous protocols. It runs up to 1.5 Mbits/sec, has on-chip baud rate generators and on-chip NRZI encoding and decoding circuits—very useful for SDLC communication. This application note shows how to write I/O drivers for the 82530 to do initialization and data links using asynchronous (ASYNC) and SDLC protocols. The appendix includes sections to show how the on-chip baud rate generators could be programmed, how the modem control pins could be used, and how the 82530 could be interfaced to INTEL's 80186/188 processors.

This article deals with the software for the following:

1. SCC port definition
2. Accessing the SCC registers
3. Initialization for ASYNC communication
4. ASYNC communication in polling mode
5. ASYNC communication in interrupt mode
6. Initialization for SDLC communication
7. SDLC frame reception
8. SDLC frame transmission
9. SDLC interrupt routines

The description is written around illustrations of the actual software written in PLM86 for a 80186 - 82530 system.

1. SCC Port Definition

The Figure 1 shows how the 4 ports (2 per channel) of the SCC can be defined. Note that the sequence of ports in the ascending order of addresses is *not* the one that is normally expected. In the ascending order it is: command (B), data (B), command (A) and data (A). In an 80186 - 82530 system, the interconnection is as follows:

	PCSn	—	CS	
	A1	—	D/C	
80186 pins	A2	—	A/B	82530 pins
	RD	—	RD	
	WR	—	WR	

2. Accessing the SCC Registers

The SCC has 16 registers on each of the channels (A and B). For each channel there is only one port, the command port, to access all the registers. The register #0 can be always accessed directly through the command port. All other registers are accessed indirectly through register #0. First, the number of the register to be accessed is written to the register #0 - see the statement, in Figure 2: 'output (ch_a_command) = reg_no and 0fh'. Then, the desired register is written to or read out. The Figure 2 shows 4 procedures: rra and wra, for reading and writing channel A registers; rrb and wrb, for reading and writing channel B registers. The read procedures are of the type 'byte' - they return the contents of the register being read. The write procedures require two parameters - the register number and the value to be written.

```
/*-------------------------------------------------------------------------*/

declare ch_b_command   literally 'pcs5 + 0', /* scc channel_b command word*/
        ch_b_data      literally 'pcs5 + 2', /* scc channel_b data word */
        ch_a_command   literally 'pcs5 + 4', /* scc channel_a command word */
        ch_a_data      literally 'pcs5 + 6'; /* scc channel_a data word */

/*-------------------------------------------------------------------------*/
```

231262-1

Figure 1. SCC Port Definition

```
/*-------------------------------------------------------------------------*/

/* read selected scc register */

rra: procedure (reg_no) byte;
     declare reg_no byte;

     if (reg_no and 0fh) <> 0
     then output(ch_a_command) = reg_no and 0fh;
     return input(ch_a_command);
end rra;

rrb: procedure (reg_no) byte;
     declare reg_no byte;

     if (reg_no and 0fh) <> 0
     then output (ch_b_command) = reg_no and 0fh;
     return input(ch_b_command);
end rrb;


/* write selected scc register */

wra: procedure (reg_no, value);
     declare reg_no byte;
     declare value byte;

     if (reg_no and 0fh) <> 0
     then output (ch_a_command) = reg_no and 0fh;
     output (ch_a_command) = value;
end wra;

wrb: procedure (reg_no, value);
     declare reg_no byte;
     declare value byte;

     if (reg_no and 0fh) <> 0
     then output (ch_b_command) = reg_no and 0fh;
     output (ch_b_command) = value;
end wrb;

/*-------------------------------------------------------------------------*/
```

231262-2

Figure 2. Accessing the SCC Registers

3. Initialization for ASYNC Operation

In the following example, channel B of the SCC is used to perform ASYNC communication. Figure 3 shows how the channel B is initialized and configured for ASYNC operation. This is done by writing the various channel B registers with the proper parameters as shown. The comments in the program show what is achieved by each statement. After a software reset of the channel, register #4 should be written before writing to the other registers. The on-chip Baud Rate Generator is used to generate a 1200 bits/sec clock for both the transmitter and the receiver. The interrupts for transmitter and/or receiver are enabled only for the interrupt mode of operation; for polling, interrupts must be kept disabled.

4. ASYNC Communication in Polling Mode

Figure 4 shows the procedures for reading in a received character from the 82530 (scc__in) and for writing out a character to the 82530 (scc__out) in the polling mode.

The scc__in procedure returns a byte value which is the character read in. The receiver is polled to find if a character has been received by the SCC. Only when a character has been received, the character is read in from the data port of the SCC channel B.

The scc__out procedure requires a byte parameter which is the character being written out. The transmit-

```
/*-------------------------------------------------------------------------*/

scc_init_b: procedure;

/* scc ch B register initialization for ASYNC mode */

      call wrb(09, 01000000b);     /* channel B reset */
      call wrb(04, 11001110b);     /* 2 stop, no parity, brf = 64x */
      call wrb(02, 00100000b);     /* vector = 20h */
      call wrb(03, 11000000b);     /* rx 8 bits/char, no auto-enable */
      call wrb(05, 01100000b);     /* tx 8 bits/char */
      call wrb(06, 00000000b);
      call wrb(07, 00000000b);
      call wrb(09, 00000001b);     /* vector includes status */
      call wrb(10, 00000000b);
      call wrb(11, 01010110b);     /* rxc = txc = BRG , trxc = BRG out */
      call wrb(12, 00011000b);     /* to generate 1200 baud, x64 @ 4 mhz */
      call wrb(13, 00000000b);
      call wrb(14, 00000011b);     /* BRG source = SYS CLK, enable BRG */
      call wrb(15, 00000000b);     /* all ext status interrupts off */

     /* enables */

      call wrb(03, 11000001b);      /* scc-b receive enable */
      call wrb(05, 11101010b);      /* scc-b transmit enable, dtr on, rts on */

     /* enable interrupts - only for interrupt driven ASYNC I/O */

      call wrb(09, 00001001b);     /* master IE, vector includes status */
      call wrb(01, 00010011b);     /* tx ,rx, ext interrupts enable */

end scc_init_b;

/*-------------------------------------------------------------------------*/
                                                              231262-3
```

Figure 3. Initialization for ASYNC Communication

```
/*-------------------------------------------------------------------------*/

/* scc data character input from channel B */

scc_in: procedure byte;

     declare char byte;

     do while (input(ch_b_command) and 1h) = 0; end;
     char = input(ch_b_data);     /* if rx data character is available */
     return char;                 /* then input it to buffer */

end scc_in;

/* scc data character output to channel B */

scc_out: procedure (char);

     declare char byte;

     do while (input(ch_b_command) and 4h) = 0; end;
     output(ch_b_data) = char;    /* if tx buff empty then transfer the */
                                  /* data character to tx buff */

end scc_out;

/*-------------------------------------------------------------------------*/
```

231262-4

Figure 4. ASYNC Communication in Polling Mode

ter is polled for being ready to transmit the next character before writing the character out to the data port of SCC channel B.

Typical calls to these procedures are:

```
abc__variable = scc__in;
call scc__out (xyz__variable);
```

5. ASYNC Communication in Interrupt Mode

In contrast to polling for the receiver and/or the transmitter to be ready with/for the next character, the 82530 can be made to interrupt when it is ready to do receive or transmit.

The on-chip interrupt controller of the SCC can be made to operate in the vectored mode. In this mode, it generates interrupt vectors that are characteristic of the event causing the interrupt. For the example here, the vector base is programmed at 20h and 'Vector Includes Status' (VIS) mode is set - WR9 = XXX0XX01. Vectors and the associated events are:

Vector	Procedure	Event Causing Interrupt
20h	txintr__b	ch__b - transmit buffer empty
22h	esi__b	ch__b - external/status change
24h	rxintr__b	ch__b - receive character available
26h	src__b	ch__b - special receive condition
28h	txintr__a	ch__a - transmit buffer empty
2ah	esi__a	ch__a - external/status change
2ch	rxintr__a	ch__a - receive character available
2eh	src__a	ch__a - special receive condition

NOTE:
Odd vector numbers do not exist.

Figure 5 shows the interrupt procedures for the channel B operating in ASYNC mode. The transmitter buffer empty interrupt occurs when the transmitter can accept one more character to output. In the interrupt procedure for transmit, the byte char__out__530 is output. Following this, is an epiloge that is *common to all the*

interrupt procedures; the first statement is an end of interrupt command to the 82530 - *note* that it is issued to *channel A* - and the second is an End of Interrupt (EOI) command to the 80186 interrupt controller which is, in fact, receiving the interrupt from the 82530.

The receive buffer full interrupt occurs when the receiver has at least one character in its buffer, waiting to be read in by the CPU.

The esi_b is not enabled to occur and src_b cannot occur in the ASYNC mode unless the receiver is overrun or a parity error occurs.

```
/*-------------------------------------------------------------------------*/

/* channel B interrupt procedures */

txintr_b:    procedure    interrupt 20h;

    output (ch_b_data) = char_out_530;

    call wra(00,38h);                /* reset highest IUS */
    output (eoir_186)  = 8000h;      /* non specific EOI */
    return;
end txintr_b;

esi_b:       procedure    interrupt 22h;

    call wrb(00,10h);                /* reset ESI */

    call wra(00,38h);                /* reset highest IUS */
    output (eoir_186)  = 8000h;      /* non specific EOI */
    return;
end esi_b;

rxintr_b:    procedure    interrupt 24h;

    char_in_530 = input (ch_b_data);

    call wra(00,38h);                /* reset highest IUS */
    output (eoir_186)  = 8000h;      /* non specific EOI */
    return;
end rxintr_b;

src_b:       procedure    interrupt 26h;

    call wrb(00,30h);                /* error reset */

    call wra(00,38h);                /* reset highest IUS */
    output (eoir_186)  = 8000h;      /* non specific EOI */
    return;
end src_b;

/*-------------------------------------------------------------------------*/
```

231262-5

Figure 5. ASYNC Communication in Interrupt Mode

6. Initialization for SDLC Communication

Channel A of the SCC is programmed for being used for SDLC operation. It uses the DMA channels on the 80186. Figure 6 shows the initialization procedure for channel A. The comments in the software show the effect of each statement. The on-chip Baud Rate Generator is used to generate a clock of 125 kHz both for reception and transmission. This procedure is just to prepare the channel A for SDLC operation. The actual transmission and reception of frames is done using the procedures described further.

7. SDLC Frame Reception

Figure 7 shows the entire set-up necessary to receive a SDLC frame. First the DMA controller is programmed with the receive buffer address (@rx__buff), byte count, mode etc and is also enabled. Then a flag indicating reception of the frame is reset. An Error Reset command is issued to clear up any pending error conditions. The receive interrupt is enabled to occur at the end of frame reception (Special Receive Condition); lastly, the receiver is enabled and put in the Hunt mode (to detect the SDLC flag). When the first flag is detected on the RxDA pin, it goes from the Hunt to the Sync mode. It receives the frame and the end of frame interrupt (src__b, vector = 2eh) occurs.

8. SDLC Frame Transmission

Figure 8 shows the procedure for transmitting a SDLC frame once channel A is initialized. The DMA controller is initialized with the transmit buffer address (@tx__buff (1)) - *note*, it is the second byte of the transmit buffer - and the byte count - again one less than the total buffer length. This is done because the first byte in the buffer is output directly using an I/O instruction and not by DMA. Then the flag indicating frame transmitted is reset. The events following are very critical in sequence:

a. Reset external status interrupts
b. Enable the transmitter
c. Reset transmit CRC
d. Enable transmitter underrun interrupt
e. Enable the DMA controller
f. Output first byte of the transmit block to data port
g. Reset Transmit Underrun Latch

2

```
/*-------------------------------------------------------------------------*/

scc_init_a: procedure;

/* scc ch A register initialization for SDLC mode */

      call wra(09, 10000000b);     /* channel A reset */
      call wra(04, 00100000b);     /* SDLC mode */
      call wra(01, 01100000b);     /* DMA for Rx */
      call wra(03, 11000000b);     /* 8 bit Rx char, Rx disable */
      call wra(05, 01100000b);     /* 8 bit Tx char, Tx disable */
      call wra(06, 01010101b);     /* node address */
      call wra(07, 01111110b);     /* SDLC flag */
      call wra(10, 10000000b);     /* preset CRC, NRZ encoding */
      call wra(11, 01010110b);     /* rxc = txc = BRG , trxc = BRG out */
      call wra(12, 00001110b);     /* to generate 125 Kbaud, x1 @ 4 mhz */
      call wra(13, 00000000b);
      call wra(14, 00000110b);     /* BRG source = SYS CLK, DMA for Tx */
      call wra(15, 00000000b);     /* all ext status interrupts off */

    /* enables */

      call wra(14, 00000111b);      /* enable : BRG */
      call wra(01, 11100000b);      /* enable : dreq */
      call wra(09, 00001001b);      /* master IE, vector includes status */

end scc_init_a;

/*-------------------------------------------------------------------------*/
                                                              231262-6
```

Figure 6. Initialization for SDLC Communication

```
/*-------------------------------------------------------------------------*/

rx_init: procedure;

     declare dma_0_mode literally '1010001001000000b';
     /* src=IO, dest=M(inc), sync=src, TC, noint, priority, byte */

     outword(dma_0_dpl) = low16(@rx_buff);
     outword(dma_0_dph) = high16(@rx_buff);
     outword(dma_0_spl) = ch_a_data;
     outword(dma_0_sph) = 0;
     outword(dma_0_tc)  = block_length + 2;        /* +2 for CRC */
     outword(dma_0_cw)  = dma_0_mode or 0006h;     /* start DMA channel 0 */

     frame_recd = 0;                  /* reset frame received flag */

     call wra(00,30h);                /* error reset */
     call wra(01, 11111001b);         /* sp. cond intr only, ext int enable */
     call wra(03, 11010001b);         /* enable receiver, enter hunt mode */

end rx_init;

/*-------------------------------------------------------------------------*/
                                                              231262-7
```

Figure 7. SDLC-DMA Frame Reception

```
/*-------------------------------------------------------------------------*/

tx_init: procedure;

     declare dma_1_mode literally '0001011010000000b';
     /* src=M(inc), dest=IO, sync=dest, TC, noint, noprior, byte */

     outword(dma_1_spl) = low16(@tx_buff(1));
     outword(dma_1_sph) = high16(@tx_buff(1));
     outword(dma_1_dpl) = ch_a_data;
     outword(dma_1_dph) = 0;
     outword(dma_1_tc)  = block_length - 1;    /* -1 for first byte */

     frame_tx = 0;                    /* reset frame transmitted flag */

     call wra(00, 00010000b);         /* reset ESI */
     call wra(05, 01101011b);         /* enable transmitter */
     call wra(00, 10101000b);         /* reset tx CRC, TxINT pending */
     call wra(15, 01000000b);         /* enable : TxU int */

     outword(dma_1_cw)  = dma_1_mode or 0006h;     /* start DMA channel 1 */
     output(ch_a_data) = tx_buff(0);     /* first byte - address field */
     call wra(00, 11000000b);            /* Reset Tx Underrun latch */

end tx_init;

/*-------------------------------------------------------------------------*/
                                                              231262-8
```

Figure 8. SDLC-DMA Frame Transmission

```
/*-------------------------------------------------------------------------*/

/* channel A interrupt procedures */

txintr_a:       procedure       interrupt 28h;

    call wra(00,38h);                   /* reset highest IUS */
    output (eoir_186)  = 8000h;         /* non specific EOI */
    return;
end txintr_a;

esi_a:          procedure       interrupt 2ah;

    call wra(00,10h);                   /* reset ESI */
    tx_stat = rra(0);                   /* read in status */
    frame_tx = 0ffh;                    /* set frame transmitted flag */

    call wra(00,38h);                   /* reset highest IUS */
    output (eoir_186)  = 8000h;         /* non specific EOI */
    return;
end esi_a;

rxintr_a:       procedure       interrupt 2ch;

    call wra(00,38h);                   /* reset highest IUS */
    output (eoir_186)  = 8000h;         /* non specific EOI */
    return;
end rxintr_a;

src_a:          procedure       interrupt 2eh;

    rx_stat = rra(1);
    call wra(00,30h);                   /* error reset */
    call wra(03,11000000b);             /* disable rx */
    frame_recd = 0ffh;                  /* set frame received flag */

    call wra(00,38h);                   /* reset highest IUS */
    output (eoir_186)  = 8000h;         /* non specific EOI */
    return;
end src_a;

/*-------------------------------------------------------------------------*/
```

231262-9

Figure 9. SDLC-DMA Interrupt Routines

The frame gets transmitted out with all bytes, except the first one, being fetched by the SCC using the DMA controller. At the end of the block the DMA controller stops supplying bytes to the SCC. This makes the transmitter underrun. Since the Transmitter Underrun Latch is in the reset state at this moment, the CRC bytes are appended by the SCC at the end of the transmit block going out. An External Status Change interrupt (esi__a, vector = 2ah) is generated with the bit for transmitter underrun set in RR0 register. This interrupt occurs when the CRC is being transmitted out and *not* when the frame is completely transmitted out.

9. SDLC Interrupt Routines

Figure 9 shows all the interrupt procedures for channel A when operating in the SDLC mode. The procedures of significance here are esi__a and src__a.

The end of frame reception results in the src__a procedure getting executed. Here the status in register RR1 is stored in a variable rx__stat for future examination. Any error bits set in status are reset, receiver is disabled and the flag indicating reception of a new frame is set.

The esi__a procedure is executed when CRC of the transmitted frame is just going out of the SCC. Reset External Status Interrupt command is executed, the external status is stored in a variable tx__stat for future examination and the flag indicating transmission of the frame is set.

End of frame processing is required after both of these interrupt procedures. It involves looking at rx__stat and tx__stat and checking if the desired operation was successful. The buffers used, may have to be recovered or new ones obtained to start another frame transmission or reception.

CONCLUSIONS

This article should ease the process of writing a complete data link driver for ASYNC and SDLC modes since most of the hardware dependent procedures are illustrated here. It was a conscious decision to make the procedures as small and easy to understand as possible. This had to be done at the expense of making the procedures general and not dealing with various exception conditions that can occur.

REFERENCES

1. 82530 Data Sheet, Order #230834-001
2. 82530 SCC Technical Manual, Order #230925-001

APPENDIX A
82530—BAUD RATE GENERATORS

The 82530 has two Baud Rate Generators (BRG) on chip—one for each channel. They are used to provide the baud rate or serial clock for receive and transmit operations. This article describes how the BRG can be programmed and used.

The BRG for each channel is totally independent of each other and have to be programmed separately for each channel. This article describes how any one of the two BRGs can be programmed for operation. To use the BRG, four steps have to be performed:

1. Determine the Baud Rate Time Constant (BRTC) to be programmed into registers WR12 (LSB) and WR13 (MSB).
2. Program in register WR11, to specify where the output of the BRG must go to.
3. Program the clock source to the BRG in register WR14.
4. Enable the BRG.

Step 1: Baud Rate Time Constant (BRTC)

The BRTC is determined by a simple formula:

$$BRTC = \frac{\text{Serial Clock Frequency}}{2 \times (\text{Baud Rate} \times \text{Baud Rate Factor})} - 2$$

Example:

For Serial Clock Frequency = 4 MHz

Baud Rate = 9600

Baud Rate Factor = 16

$$BRTC = \frac{4000000}{2 \times (9600 \times 16)} - 2$$

$$= 13.021 - 2 = 11.021$$

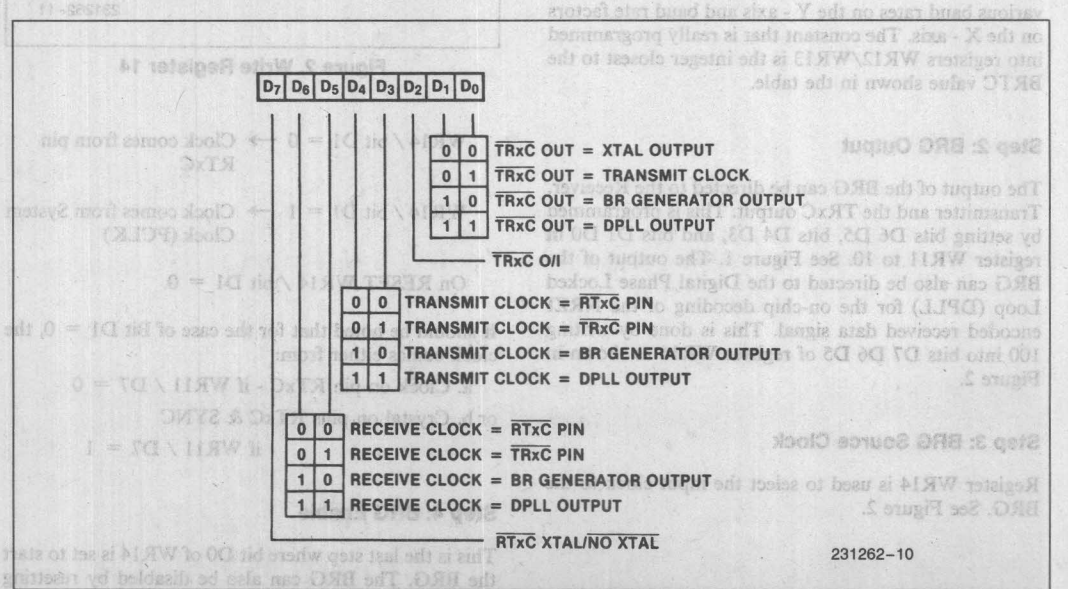


Figure 1. Write Register 11

Table 1. BRTC - Baud Rate Time Constant

		Baud Rate Factor			
		1	16	32	64
Baud Rate	9600	206.333	11.021	4.510	1.255
	4800	414.667	24.042	11.021	4.510
	2400	831.333	50.083	24.042	11.021
	1200	1664.667	102.167	50.083	24.042
	600	3331.333	206.333	102.167	50.083
	300	6664.667	414.667	206.333	102.167

Since only integers can be written into the registers WR12/WR13 this will have to be rounded off to 11 and it will result in an error of:

$$\frac{\text{fraction}}{\text{BRTC}} \times 100 = \frac{0.021}{11.021} \times 100 = 0.19\%$$

This error indicates that the baud rate signal generated by the BRG does not provide the exact frequency required by the system. This error is more serious for smaller baud rate factors. For asynchronous systems, errors up to 5% are considered acceptable.

Note that for BRTC = 0, BRG output frequency = 1/4 × Serial Clock Freq.

Table 1 shows the BRTC for a 4 MHz serial clock with various baud rates on the Y - axis and baud rate factors on the X - axis. The constant that is really programmed into registers WR12/WR13 is the integer closest to the BRTC value shown in the table.

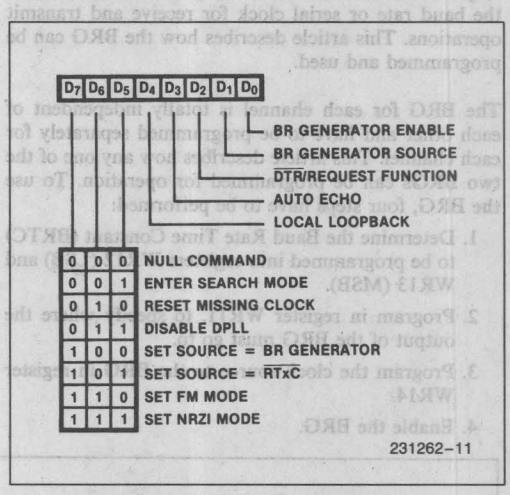


Figure 2. Write Register 14

Step 2: BRG Output

The output of the BRG can be directed to the Receiver, Transmitter and the TRxC output. This is programmed by setting bits D6 D5, bits D4 D3, and bits D1 D0 in register WR11 to 10. See Figure 1. The output of the BRG can also be directed to the Digital Phase Locked Loop (DPLL) for the on-chip decoding of the NRZI encoded received data signal. This is done by writing 100 into bits D7 D6 D5 of register WR14 as shown in Figure 2.

Step 3: BRG Source Clock

Register WR14 is used to select the input clock to the BRG. See Figure 2.

WR14 / bit D1 = 0 → Clock comes from pin RTxC

WR14 / bit D1 = 1 → Clock comes from System Clock (PCLK)

On RESET WR14 / bit D1 = 0.

It should be noted that for the case of Bit D1 = 0, the clock comes either from:

a. Clock on pin RTxC - if WR11 / D7 = 0

or b. Crystal on pins RTxC & SYNC - if WR11 / D7 = 1

Step 4: BRG Enable

This is the last step where bit D0 of WR14 is set to start the BRG. The BRG can also be disabled by resetting this bit.

APPENDIX B
MODEM CONTROL PINS ON THE 82530

Introduction

This article describes how the $\overline{CTS}$ and $\overline{CD}$ pins on the 82530 behave and how to write software to service these pins. The article explains when the External Status Interrupt occurs and how and when to issue the Reset External/Status Interrupt command to reliably determine the state of these pins.

Bits D3 and D5 of register RR0 show the *inverted* state of logic levels on $\overline{CD}$ and $\overline{CTS}$ pins respectively. It is important to note that the register RR0 does *not* always reflect the *current* state of the $\overline{CD}$ and $\overline{CTS}$ pins. Whenever a Reset External/Status Interrupt (RESI) command is issued, the (inverted) states of the $\overline{CD}$ and the $\overline{CTS}$ pins get updated and latched into the RR0 register and the register RR0 then reflect the inverted state of the $\overline{CD}$ and $\overline{CTS}$ pins at the time of the write operation to the chip. On channel or chip reset, the inverted state of $\overline{CD}$ and $\overline{CTS}$ pins get latched into RR0 register.

Normally, a transition on any of the pins does not necessarily change the corresponding bit(s) in RR0. In certain situations it does and in some cases it does not. A sure way of knowing the current state of the pins is to read the register RR0 *after* a RESI command.

There are two cases:

I. External/Status Interrupt (ESI) enabled.

II. Polling (ESI disabled).

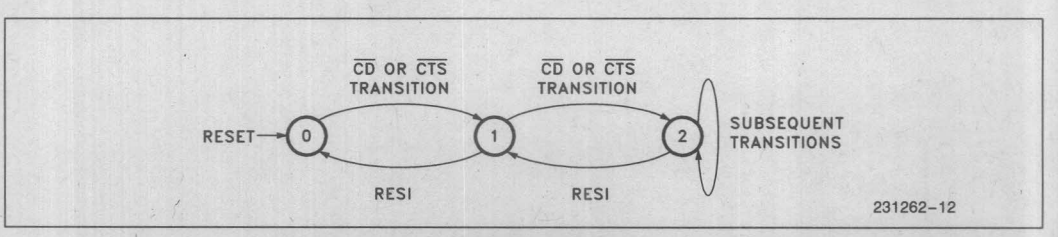
Case I: External Status Interrupt (ESI) Enabled

Whenever ESI is enabled, an interrupt can occur whenever there is a transition on $\overline{CD}$ or $\overline{CTS}$ pins - the IE bits for $\overline{CD}$ and/or $\overline{CTS}$ must also be set in WR15 for the interrupt to be enabled.

In this case, the first transition on any of these pins will cause an interrupt to occur and the corresponding bit in RR0 to change (even without the RESI command). A RESI command resets the interrupt line and also latches in the current state of both the $\overline{CD}$ and the $\overline{CTS}$ pins. If there was just one transition the RESI does not really change the contents of RR0.

If there are more than one transitions, either on the same pin or one each on both pins or multiple on both pins, the interrupt would get activated on the first transition and stay active. The bit in RR0 corresponding only to the very first transition is changed. All subsequent transitions have no effect on RR0. The first transition, in effect, freezes all changes in RR0. The first RESI command, as could be expected, latches the final (inverted) state of the $\overline{CD}$ and $\overline{CTS}$ pins into the RR0 register. Note that all the intermediate transitions on the pins are lost (because the response to the interrupt was not fast enough). The interrupt line gets reset for only a brief moment following the first RESI command. This brief moment is approximately 500 ns for the 82530. After that the interrupt becomes active again. A second RESI command is necessary to reset the interrupt. Two RESI commands resets the interrupt line independent of the number of transitions occurred.

Whenever operating with ESI enabled, it is recommendable to issue two back-to-back RESI commands and then read the RR0 register to reliably determine the state of the $\overline{CD}$ and $\overline{CTS}$ pins and also to reset the interrupt line in case multiple transitions may have occurred.



State Diagram

Case II: Polling RR0 for $\overline{CD}$ and $\overline{CTS}$ Pins

If RR0 is polled for determining the state of the $\overline{CD}$ and $\overline{CTS}$ pins, then the External/Status Interrupt (ESI) is kept disabled. In this case the bits in RR0 may not change even for the first transition. The best way to handle this case to always issue a RESI command before reading in the RR0 register to determine the state of $\overline{CD}$ and $\overline{CTS}$ pins. Note, however, if two back-to-back RESI commands were to be issued every time before reading in the RR0 register, the first subsequent transition will change the corresponding bit in RR0.

The state diagram above illustrates how each transition on $\overline{CD}$ and $\overline{CTS}$ pins affect the 82530 and what effect the RESI command has.

State 0

It is entered on reset. No ESI due to $\overline{CTS}$ or $\overline{CD}$ are pending in this state. Any transition on $\overline{CTS}$ or $\overline{CD}$ pins lead to the state 1 *accompanied by an immediate change in the RR0 register.*

State 1

Interrupt is active (if enabled). If a RESI command is issued, state 0 is reached where interrupt is again inactive. However, a further transition on $\overline{CTS}$ or $\overline{CD}$ pin leads to state 2 *without an immediate change in RR0 register.*

State 2

Interrupt is active (if enabled). Any further transitions have no effect. A RESI command leads to state 1, temporarily making the interrupt inactive.

CONCLUSIONS

Register RR0 does not always reflect the current (inverted) state of the $\overline{CD}$ and $\overline{CTS}$ pins. The most reliable way to determine the state of the pins in interrupt or polling mode is to issue two back-to-back RESI commands and then read RR0. While polling, the second RESI is redundant but harmless. When issuing the back-to-back RESI commands to 82530 note that the separation between the two write cycles should be at least 6 CLK + 200 ns; otherwise the second RESI will be ignored.

APPENDIX C
THE 82530 SCC - 80186 INTERFACE AP BRIEF

INTRODUCTION

The object of this document is to give the 82530 system designer an in-depth worst case design analysis of the typical interface to a 80186 based system. This document has been revised to include the new specifications for the 6 MHz 82530. The new specifications yield better margins and a 1 wait state interface to the CPU (2 wait states are required for DMA cycles). These new specifications will appear in the 1987 data sheet and advanced specification information can be obtained from your local Intel sales office. The following analysis includes a discussion of how the interface TTL is utilized to meet the timing requirements of the 80186 and the 82530. In addition, several optional interface configurations are also considered.

INTERFACE OVERVIEW

The 82530 - 80186 interface requires the TTL circuitry illustrated in Figure 1. Using five 14 pin TTL packages, 74LS74, 74AS74, 74AS08, 74AS04, and 74LS32, the following operational modes are supported:

- Polled
- Interrupt in vectored mode
- Interrupt in non-vectored mode
- Half-duplex DMA on both channels
- Full-duplex DMA on channel A

A brief description of the interface functional requirements during the five possible BUS operations follows below.

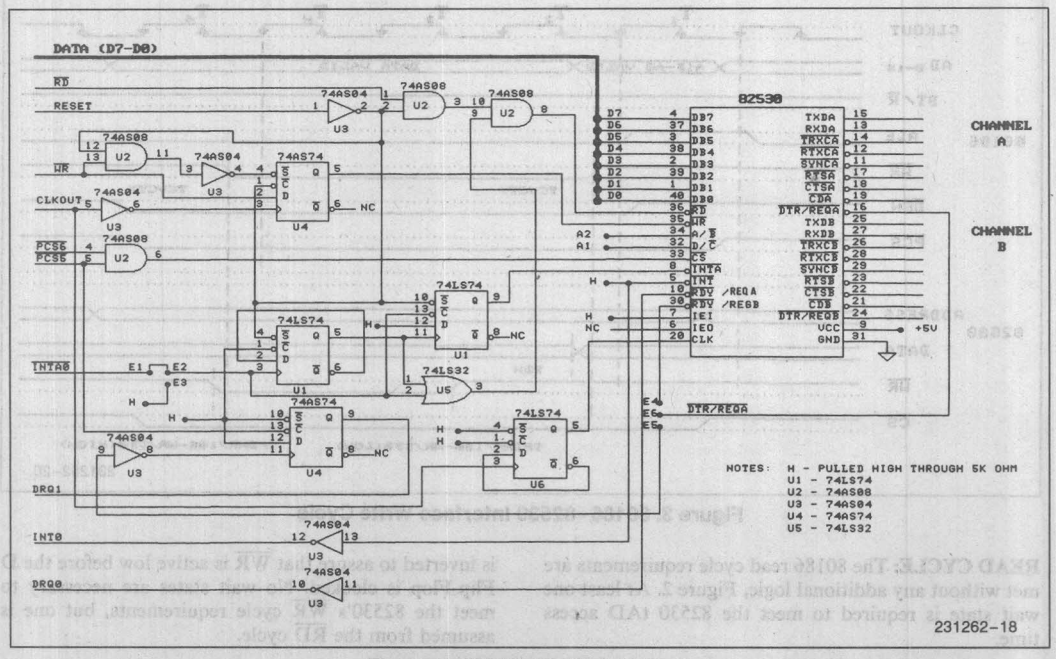


Figure 1. 82530–80186 Interface

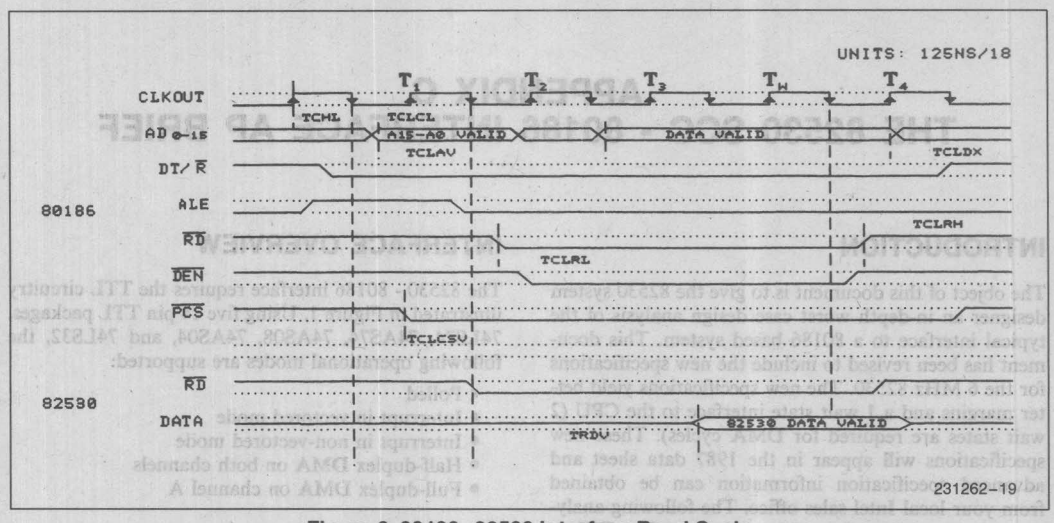


Figure 2. 80186–82530 Interface Read Cycle

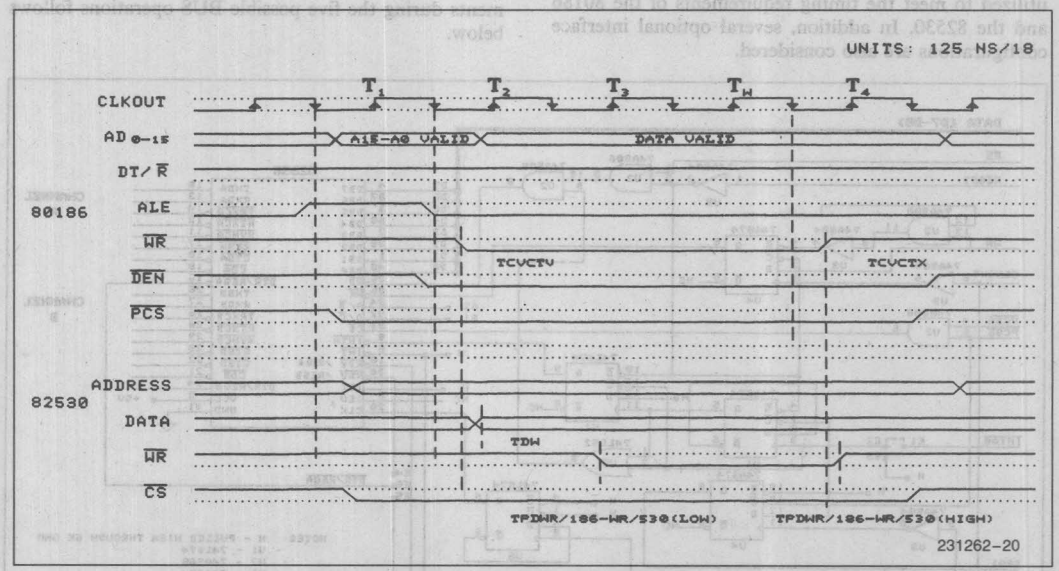


Figure 3. 80186–82530 Interface Write Cycle

READ CYCLE: The 80186 read cycle requirements are met without any additional logic, Figure 2. At least one wait state is required to meet the 82530 tAD access time.

WRITE CYCLE: The 82530 requires that data must be valid while the $\overline{WR}$ pulse is low, Figure 3. A D Flip-Flop delays the leading edge of $\overline{WR}$ until the falling edge of CLOCKOUT when data is guaranteed valid and $\overline{WR}$ is guaranteed active. The CLOCKOUT signal is inverted to assure that $\overline{WR}$ is active low before the D Flip-Flop is clocked. No wait states are necessary to meet the 82530's $\overline{WR}$ cycle requirements, but one is assumed from the $\overline{RD}$ cycle.

INTA CYCLE: During an interrupt acknowledge cycle, the 80186 provides two $\overline{INTA}$ pulses, one per bus cycle, separated by two idle states. The 82530 expects only one long $\overline{INTA}$ pulse with a $\overline{RD}$ pulse occurring only after the 82530 $\overline{IEI}/\overline{IEO}$ daisy chain settles. As

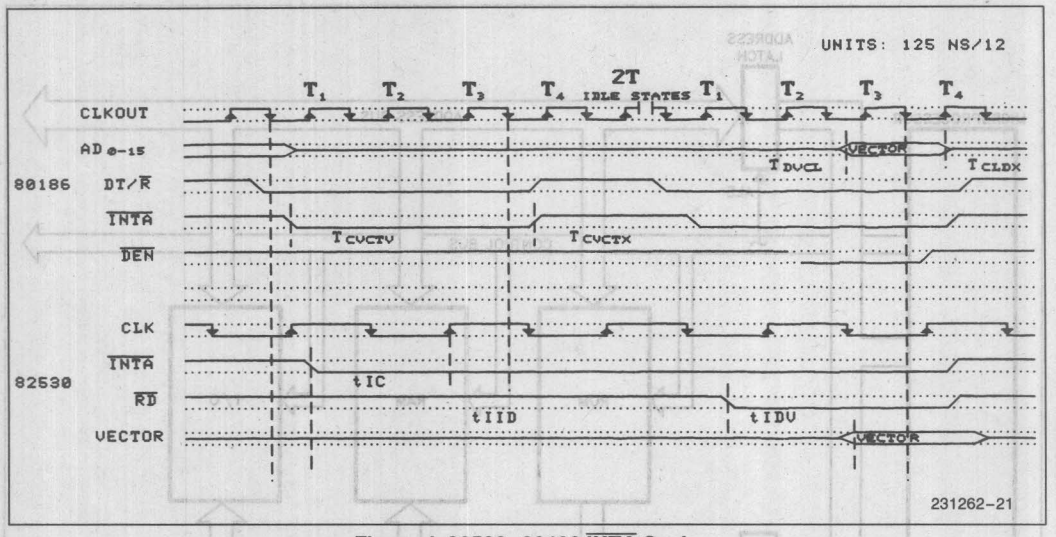


Figure 4. 82530–80186 INTA Cycle

illustrated in Figure 4, the INTA signal is sampled on the rising edge of CLK (82530). Two D Flip-Flops and two TTL gates, U2 and U5, are implemented to generate the proper INTA and RD pulses. Also, the INT signal is passively pulled high, through a 1 k resistor, and inverted through U3 to meet the 80186's active high requirement.

DMA CYCLE: Conveniently, the 80186 DMA cycle timings are the same as generic read and write operations. Therefore, with two wait states, only two modifications to the DMA request signals are necessary. First, the RDYREQA signal is inverted through U3 similar to the INT signal, and second the DTR/REQA signal is conditioned through a D Flip-Flop to prevent inadvertent back to back DMA cycles. Because the 82530 DTR/REQA signal remains active low for over five CLK (82530)'s, an additional DMA cycle could occur. This uncertain condition is corrected when U4 resets the DTR/REQ signal inactive high. Full Duplex on both DMA channels can easily be supported with one extra D Flip-Flop and an inverter.

RESET: The 82530 does not have a dedicated RESET input. Instead, the simultaneous assertion of both RD and WR causes a hardware reset. This hardware reset is implemented through U2, U3, and U4.

ALTERNATIVE INTERFACE CONFIGURATIONS

Due to its wide range of applications, the 82530 interface can have many varying configurations. In most of these applications the supported modes of operation need not be as extensive as the typical interface used in this analysis. Two alternative configurations are discussed below.

8288 BUS CONTROLLER: An 80186 based system implementing an 8288 bus controller will not require the preconditioning of the WR signal through the D Flip-Flop U4. When utilizing an 8288, the control signal IOWC does not go active until data is valid, therefore, meeting the timing requirements of the 82530. In such a configuration, it will be necessary to logically OR the IOWC with reset to accommodate a hardware reset operation.

NON-VECTORED INTERRUPTS: If the 82530 is to be operated in the non-vectored interrupt mode (B step only), the interface will not require U1 or U5. Instead, INTA on the 82530 should be pulled high, and pin 3 of U2 (RD AND RESET) should be fed directly into the RD input of the SCC.

Obviously, the amount of required interface logic is application dependent and in many cases can be considerably less than required by the typical configuration, supporting all modes of SCC operation.

DESIGN ANALYSIS

This design analysis is for a typical microprocessor system, pictured in Figure 5. The Timing analysis assumes an 8 MHz 80186 and a 6 MHz 82530 being clocked at 4 MHz. The 4 MHz clock is the 80186 CLKOUT divided by two by a flip-flop (U6). Also, included in the analysis are bus loading, and TTL-MOS compatibility considerations.

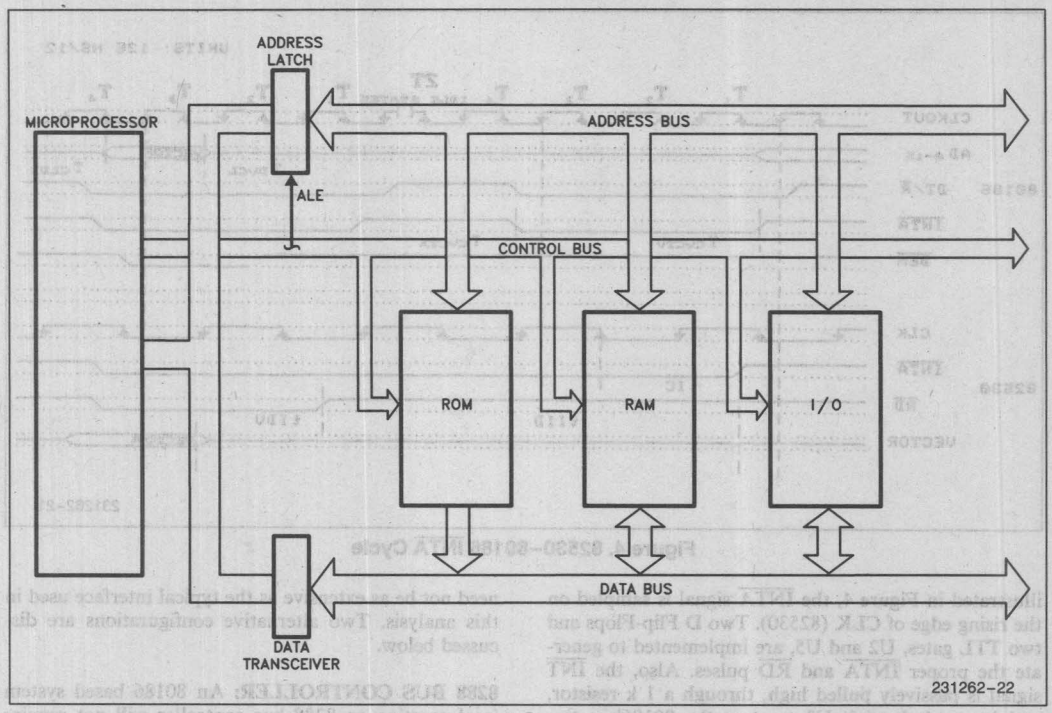


Figure 5. Typical Microprocessor System

Bus Loading and Voltage Level Compatabilities

The data and address lines do not exceed the drive capability of either 80186 or the 82530. There are several control lines that drive more than one TTL equivalent input. The drive capability of these lines are detailed below.

$\overline{WR}$: The $\overline{WR}$ signal drives U3 and U4.

* Iol (2.0 mA) > Iil (−0.4 mA + −0.5 mA)
Ioh (−400 μA) > Iih (20 μA + 20 μA)

$\overline{PCS5}$: The $\overline{PCS5}$ signal drives U2 and U4.

* Iol (2.0 mA) > Iil (−0.5 mA + −0.5 mA)
Ioh (−400 μA) > Iih (20 μA + 20 μA)

$\overline{INTA}$: The $\overline{INTA}$ signal drives 2(U1) and U5.

* Iol (2.0 mA) > Iil (−0.4 mA + −0.8 mA + −0.4 mA)
Ioh (−400 μA) > Iih (20 μA + 40 μA + 20 μA)

All the 82530 I/O pins are TTL voltage level compatible.

TIMING ANALYSIS

Certain symbolic conventions are adhered to throughout the analysis below and are introduced for clarity.

1. All timing variables with a lower case first letter are 82530 timing requirements or responses (i.e., tRR).
2. All timing variables with Upper case first letters are 80186 timing responses or requirements unless preceded by another device's alpha-numeric code (i.e., Tclcl or '373 Tpd).
3. In the write cycle analysis, the timing variable Tpd$\overline{WR}$186-$\overline{WR}$530 represents the propagation delay between the leading or trailing edge of the $\overline{WR}$ signal leaving the 80186 and the $\overline{WR}$ edge arrival at the 82530 $\overline{WR}$ input.

Read Cycle

1. **tAR:** Address valid to $\overline{RD}$ active set up time for the 82530. Since the propagation delay is the worst case path in the assumed typical system, the margin is calculated only for a propagation delay constrained and not an ALE limited path. The spec value is 0 ns minimum.

* 1 Tclcl − Tclav(max) − '245 Tpd(max) + Tclrl(min) + 2(U2) Tpd(min) − tAR(min)

= 125 − 55 − 20.8 + 10 + 2(2) − 0 = 63.2 ns margin

2. **tRA:** Address to $\overline{RD}$ inactive hold time. The ALE delay is the worst case path and the 82530 requires 0 ns minimum.

* 1 Tclcl − Tclrh (max) + Tchlh(min) + '373 LE Tpd(min) − 2(U2) Tpd(max)

= 55 − 55 + 5 + 8 − 2(5.5) = 2 ns margin

3. **tCLR:** $\overline{CS}$ active low to $\overline{RD}$ active low set up time. The 82530 spec value is 0 ns minimum.

* 1 Tclcl − Tclcsv(max) − Tclrl(min) − U2 skew($\overline{RD}$ − $\overline{CS}$) + U2 Tpd(min)

= 125 − 66 − 10 − 1 + 2 = 50 ns margin

4. **tRCS:** $\overline{RD}$ inactive to $\overline{CS}$ inactive hold time. The 82530 spec calls for 0 ns minimum.

* Tcscsx(min) − U2 skew($\overline{RD}$ − $\overline{CS}$) − U2 Tpd(max)

= 35 − 1 − 5.5 = 28.5 ns margin

5. **tCHR:** $\overline{CS}$ inactive to $\overline{RD}$ active set up time. The 82530 requires 5 ns minimum.

* 1 Tclcl + 1 Tchcl − Tchcsx(max) + Tclrl(min) − U2 skew ($\overline{RD}$ − $\overline{CS}$) + U2 Tpd(min) − tCHR

= 125 + 55 − 35 − 10 − 1 + 2 − 5 = 131 ns margin

6. **tRR:** $\overline{RD}$ pulse active low time. One 80186 wait state is included to meet the 150 ns minimum timing requirements of the 82530.

* Trlrh(min) + 1($\overline{Tclcl}$wait state) − 2(U2 skew) − tRR

= (250−50) + 1(125) − 2(1) − 150 = 173 ns margin

7. tRDV: $\overline{RD}$ active low to data valid maximum delay for 80186 read data set up time (Tdvcl = 20 ns). The margin is calculated on the Propagation delay path (worst case).

* 2 Tclcl + 1($\overline{Tclcl}$wait state) − Tclrl(max) − Tdvcl(min) − '245 Tpd(max) − 82530 tRDV(max) − 2(U2) Tpd(max)

= 2(125) + 1(125) − 70 − 20 − 14.2 − 105 − 2(5.5) = 154 ns margin

8. **tDF:** $\overline{RD}$ inactive to data output float delay. The margin is calculated to $\overline{DEN}$ active low of next cycle.

* 2 Tclcl + Tclch(min) − Tclrh(max) + Tchctv(min) − 2(U2) Tpd(max) − 82530 tDF(max)

= 250 + 55 −55 + 10 − 11 − 70 = 179 ns margin

9. **tAD:** Address required valid to read data valid maximum delay. The 82530 spec value is 325 ns maximum.

* 3 Tclcl + 1($\overline{Tclcl}$wait state) − Tclav(max) − '373 Tpd(max) − '245 Tpd − Tdvcl(min) − tAD

= 375 + 125 − 55 − 20.8 −14.2 − 20 −325 = 65 ns margin

Write Cycle

1. **tAW:** Address required valid to $\overline{WR}$ active low set up time. The 82530 spec is 0 ns minimum.

* Tclcl − Tclav(max) − Tcvctv(min) − '373 Tpd(max) + Tpd$\overline{WR}$186 − $\overline{WR}$530(LOW) [Tclcl − Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)] − tAW

= 125 − 55 − 5 − 20.8 + [125 − 5 + 1 + 4.4] − 0 = 170.6 ns margin

2. **tWA:** $\overline{WR}$ inactive to address invalid hold time. The 82530 spec is 0 ns.

* Tclch(min) − Tcvctx(max) + Tchlh(min) + '373 LE Tpd(min) − Tpd$\overline{WR}$186=$\overline{WR}$530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)]

= 55 − 55 + 5 + 8 − [5.5 + 3 + 7.1] = −2.6 ns margin

3. **tCLW:** Chip select active low to $\overline{WR}$ active low hold time. The 82530 spec is 0 ns.

* 1 Tclcl − Tclcsv(max) + Tcvctv(min) − U2 Tpd(max) + Tpd$\overline{WR}$186=$\overline{WR}$530(LOW) [Tclcl − Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)]

= 125 − 66 + 5 − 5.5 + [125 − 5 + 1 + 4.4] = 183.9 ns margin

4. **tWCS:** $\overline{WR}$ invalid to Chip Select invalid hold time. 82530 spec is 0 ns.

* Tcxcsx(min) − U2 Tpd(max) − Tpd$\overline{WR}$186=$\overline{WR}$530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)]

= 35 + 1.5 − [5.5 + 3 + 7.1] = 20.9 ns margin

5. **tCHW:** Chip Select inactive high to $\overline{WR}$ active low set up time. The 82530 spec is 5 ns.

* 1 Tclcl + Tchcl(min) + Tcvctv(min) − Tchcsx(max) − U2 Tpd(max) + Tpd$\overline{WR}$186=$\overline{WR}$530(LOW) [Tclcl − Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)] − tCHW

= 125 + 55 + 5 − 35 − 5.5 + [125 −5 + 1 + 4.4] − 5 = 264 ns margin

6. **tWW:** $\overline{WR}$ active low pulse. 82530 requires a minimum of 60 ns from the falling to the rising edge of $\overline{WR}$. This includes one wait state.

2

* Twlwh [2Tclcl − 40] + 1 (Tclclwait state) − TpdWR/186−WR530(LOW) [Tclcl − Tcvctv(min) + U3 Tpd(max) + U4 Tpd(max)] + TpdWR/186=WR/530(HIGH) [U2 Tpd(min) U3 Tpd(min) + U4 Tpd(min)] − tWW

= 210 + 1(125) − [125 − 5 + 4.5 + 9.2] − [1.5 + 1 + 3.2] − 60 = 135.6 ns margin

7. **tDW:** Data valid to WR active low setup time. The 82530 spec requires 0 ns.

* Tcvctv(min) − Tcldv(max) − '245 Tpd(max) + TpdWR186−WR530(LOW) [Tclcl − Tcvctv(min) + U3 Tpd(min) + U4 Tpd(min)]

= 5 − 44 − 14.2 + 125 − 5 + 1.0 + 4.4 = 72.2 ns margin

8. **tWD:** Data valid to WR inactive high hold time. The 82530 requires a hold time of 0 ns.

* Tclch − skew {Tcvctx(max) + Tcvctx(min)} + '245 OE Tpd(min) − TpdWR186−WR530(HIGH) [U2 Tpd(max) + U3 Tpd(max) + U4 Tpd(max)]

= 55 − 5 + 11.25 − [5.5 + 3.0 + 7.1] = −50.6 ns margin

INTA Cycle:

1. **tIC:** This 82530 spec implies that the INTA signal is latched internally on the rising edge of CLK (82530). Therefore the maximum delay between the 80186 asserting INTA active low or inactive high and the 82530 internally recognizing the new state of INTA is the propagation delay through U1 plus the 82530 CLK period.

* U1 Tpd(max) + 82530 CLK period

= 45 + 250 = 295 ns

2. **tCI:** rising edge of CLK to INTA hold time. This spec requires that the state of INTA remains constant for 100 ns after the rising edge of CLK. If this spec is violated any change in the state of INTA may not be internally latched in the 82530. tCI becomes critical at the end of an INTA cycle when INTA goes inactive. When calculating margins with tCI, an extra 82530 CLK period must be added to the INTA inactive delay.

3. **tIW:** INTA inactive high to WR active low minimum setup time. The spec pertains only to 82530 WR cycle and has a value of 55 ns. The margin is calculated assuming an 82530 WR cycle occurs immediately after an INTA cycle. Since the CPU cycles following an 82530 INTA cycle are devoted to locating and executing the proper interrupt service routine, this condition should never exist. 82530 drivers should insure that at least one CPU cycle separates INTA and WR or RD cycles.

4. **tWI:** WR inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

* Tclcl − Tcvctx(max) − TpdWR186 − WR530(HIGH) [U3 Tpd(max) + U4 Tpd(max)] + Tcvctv(min) + U1 Tpd(min)

= 125 − 55 − [5.5 + 3 + 7.1] + 5 + 10 = 69.4 ns margin

5. **tIR:** INTA inactive high to RD active low minimum setup time. This spec pertains only to 82530 RD cycles and has a value of 55 ns. The margin is calculated in the same manner as tIW.

6. **tRI:** RD inactive high to INTA active low minimum hold time. The spec is 0 ns and the margin assumes CLK coincident with INTA.

* Tclcl − Tclrh(max) − 2 U2 Tpd(max) + Tcvctv(min) + U1 Tpd(min)

= 125 − 55 − 2(5.5) + 5 + 10 = 74 ns margin

7. **tIID:** INTA active low to RD active low minimum setup time. This parameter is system dependent. For any SCC in the daisy chain, tIID must be greater than the sum of tCEQ for the highest priority device in the daisy chain, tEI for this particular SCC, and tEIEO for each device separating them in the daisy chain. The typical system with only 1 SCC requires tIID to be greater than tCEQ. Since tEI occurs coincidently with tCEQ and it is smaller it can be neglected. Additionally, tEIEO does not have any relevance to a system with only one SCC. Therefore tIID > tCEQ = 250 ns.

* 4 Tclcl + 2 Tidle states − Tcvctv(max) − tIC [U1 Tpd(max) + 82530 CLK period] + Tcvctv(min) + U5 Tpd(min) + U2 Tpd(min) − tIID

= 500 + 250 − 70 − [45 + 250] + 5 + 6 + 2 − 250 = 148 ns margin

8. **tIDV:** RD active low to interrupt vector valid delay. The 80186 expects the interrupt vector to be valid on the data bus a minimum of 20 ns before T4 of the second acknowledge cycle (Tdvcl). tIDV spec is 100 ns maximum.

* 3 Tclcl − Tcvctv(max) − U5 Tpd(max) − U2 Tpd(max) − tIDV(max) − '245 Tpd(max) − Tdvcl(min)

= 375 − 70 − 25 − 5.5 − 100 − 14.2 − 20 = 140.3 ns margin

9. **tII:** $\overline{RD}$ pulse low time. The 82530 requires a minimum of 125 ns.

* 3 TclcI − Tcvctv(max) − U5 Tpd(max) − U2 Tpd(max) + Tcvctx(min) + U5 Tpd(min) + U2 Tpd(min) − tII(min)

= 375 − 70 − 25 − 5.5 + 5 + 6 + 1.5 − 125 = 162 ns margin

DMA Cycle

Fortunately, the 80186 DMA controller emulates CPU read and write cycle operation during DMA transfers. The DMA transfer timings are satisfied using the above analysis. Because of the 80186 DMA request input requirements, two wait states are necessary to prevent inadvertent DMA cycles. There are also CPUDMA intracycle timing considerations that need to be addressed.

1. **tDRD:** $\overline{RD}$ inactive high to $\overline{DTRREQ}$ (REQUEST) inactive high delay. Unlike the $\overline{READYREQ}$ signal, $\overline{DTRREQ}$ does not immediately go inactive after the requested DMA transfer begins. Instead, the $\overline{DTRREQ}$ remains active for a maximum of 5 tCY + 300 ns. This delayed request pulse could trigger a second DMA transfer. To avoid this undesirable condition, a D Flip Flop is implemented to reset the $\overline{DTRREQ}$ signal inactive low following the initiation of the requested DMA transfer. To determine if back to back DMA transfers are required in a source synchronized configuration, the 80186 DMA controller samples the service request line 25 ns before T1 of the deposit cycle, the second cycle of the transfer.

* 4 TclcI − Tclcsv(max) − U4Tpd(max) − Tdrqcl(min)

= 500 − 66 − 10.5 − 25 = 398.5 ns margin

2. tRRI: 82530 $\overline{RD}$ active low to $\overline{REQ}$ inactive high delay. Assuming source synchronized DMA transfer, the 80186 requires only one wait state to meet the tRRI spec of 200 ns. Two are included for consistency with tWRI.

* 2 TclcI + 2($\overline{TclcI}$wait state) − TclrI(max) − 2(U2) Tpd(max) − Tdrqcl − tRRI

=2(125) + 2(125) − 70 − 2(5.5) − 200 = 219 ns margin

3. tWRI: 82530 $\overline{WR}$ active low to $\overline{REQ}$ inactive high delay. Assuming destination synchronized DMA transfers, the 80186 needs two wait states to meet the tWRI spec. This is because the 80186 DMA controller samples requests two clocks before the end of the deposit cycle. This leaves only 1 TclcI + n(wait states) minus $\overline{WR}$ active delay for the 82530 to inactivate its $\overline{REQ}$ signal.

* TclcI + 2($\overline{TclcI}$wait state) − Tcvctv(min) − Tpd$\overline{WR}$186−$\overline{WR}$530(LOW) [TclcI − Tcvctv(min) + U3 Tpd(max) + U4 Tpd(max)] − Tdrqcl − tWRI

=375 − 5 − [125 − 5 + 4.5 + 9.2] − 25 − 200 = 11.3 ns margin

NOTE:
If one wait state DMA interface is required, external logic, like that used on the $\overline{DTRREQ}$ signal, can be used to force the 82530 $\overline{REQ}$ signal inactive.

4. **tREC:** CLK recovery time. Due to the internal data path, a recovery period is required between SCC bus transactions to resolve metastable conditions internal to the SCC. The DMA request lines are masked from requesting service until after the tREC has elapsed. In addition, the CPU should not be allowed to violate this recovery period when interleaving DMA transfers and CPU bus cycles. Software drivers or external logic should orchestrate the CPU and DMA controller operation to prevent tREC violation. In this example circuit, tREC could be improved by clocking the '530 with a 6 MHz clock.

Reset Operation

During hardware reset, the system RESET signal is asserted high for a minimum of four 80186 clock cycles (1000 ns). The 82530 requires $\overline{WR}$ and $\overline{RD}$ to be simultaneously asserted low for a minimum of 250 ns.

* 4 TclcI − U3 Tpd(max) − 2(U2) Tpd(max) + U4 Tpd(min) − tREC

= 1000 − 17.5 − 2(5.5) + 3.5 − 250 ns = 725 ns margin

82530 VALID ACCESS LOGIC

Due to the unique internal data path of the 82530, an intra-access recovery time must be provided to settle any internal metastable conditions. This internal metastble condition gives rise to the Clock Recovery {tREC] specification required by the 82530. This tREC is measured from the rising edge of a $\overline{RD}$ or $\overline{WR}$ to the falling edge of the next $\overline{RD}$ or $\overline{WR}$ intended for the 82530, and equates to 6 CLK's + 130 ns. Effectively, this specification implies that the system must provide 1130 ns (6 MHz 82530) between every CPU or other DMA access to the 82530. (Figure 1.)

Systems that only allow CPU access to the 82530 are not significantly impacted by this clock recovery time. In CPU access only designs, the software designer can insert NOP's to guarantee the tREC idle time in between successive CPU $\overline{RD}$ or $\overline{WR}$ cycles to the 82530. Unfortunately, systems that contain more than one direct memory access device, interfacing with the 82530, will require external hardware to arbitrate 82530 accesses and thereby guaranteeing the tREC restriction.

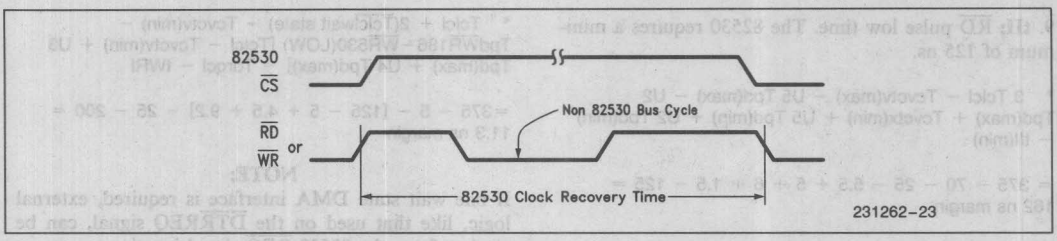


Figure 6

EXTERNAL VALID ACCESS HARDWARE

To accommodate this clock recovery specification, external hardware has been designed for the 82530 systems containing several DMA devices accessing the 82530 (ie., a CPU and a DMA controller). This logic has been tailored for an 80186 environment but can easily be modified to fit 8086 or 80286 systems.

LOGIC STATE MACHINE

There are two basic functions that need to be performed by the external logic. The first is to mask the $\overline{CS}$ signal from reaching the 82530 until the tREC intra-access idle time has elapsed. The second task is to generate a not ready condition to the CPU or DMA device until the tREC period has expired and the minimum wait state requirement for the particular access has been satisfied. The simple state machine, Figure 7, illustrates the required operation.

The TTL logic pictured in Figure 7 implements the state machine with some assorted gates, a flip-flop, and a shift register. $\overline{PCS}$ from the 80186 should be qualified with $\overline{RD}$ + $\overline{WR}$ to eliminate switching glitches during T1. The 'LS74 and 'LS00 perform rising edge detection to reset the shift register. The shift register clocks out the tREC period to enable $\overline{CS}$ and the additional 2 CLK's {82530} to satisfy the 82530 3 wait state requirement. The 80186 should be programmed to use the internal wait state generator {3 wait states for the 82530 and an 8 MHz 80186} and the external READY signal.

Note of caution: This hardware logic has not been verified on a bread board in an actual system. The hardware designer should verify that this logic fulfills his particular system timing requirements.

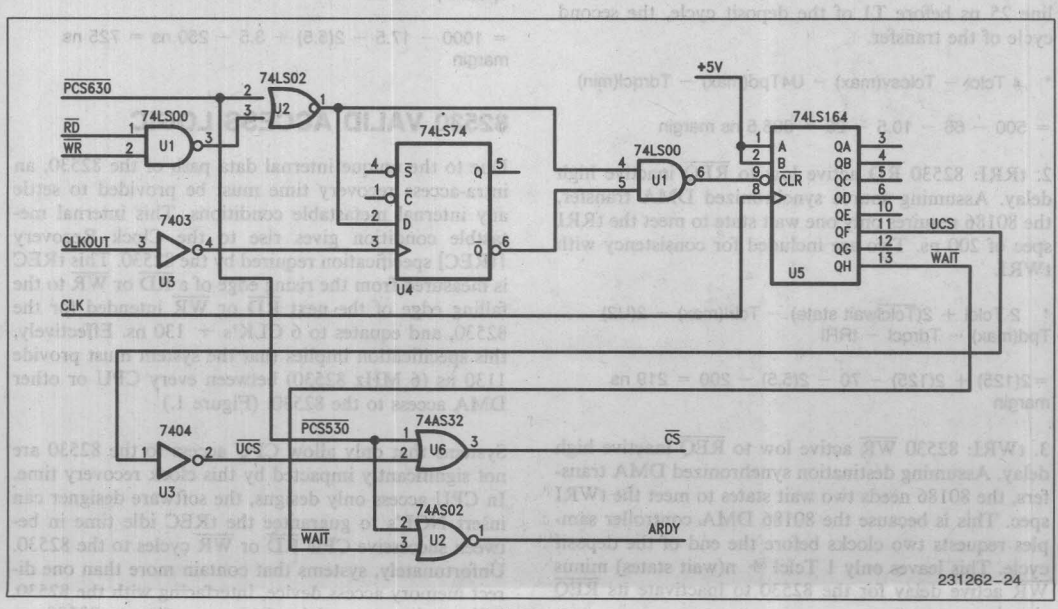


Figure 7

Other Components

3

3 Other Components

- **Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions**
- **Selectable Interrupts**
- **On-Chip Primary and Secondary Address Recognition**
- **Automatic Handling of Addressing and Handshake Protocol**
- **Provision for Software Implementation of Additional Features**
- **Either as a Major or a Minor Talker/ Listener with Primary or Secondary Addressing**
- **DMA Handshake Provision Allows for Bus Transfers without CPU Intervention**
- **Trigger Output Pin**
- **On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers**

3

The 8291A is an enhanced version of the 8291 GPIB Talker/Listener designed to interface microprocessors to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller. The controller function can be added with the 8292 GPIB Controller, and the 8293 GPIB Transceiver performs the electrical interface for Talker/Listener and Talker/Listener/Controller configurations.

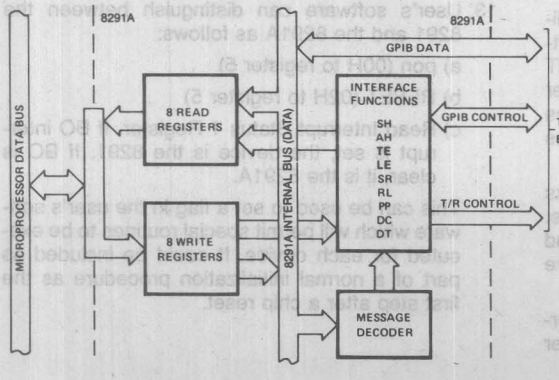
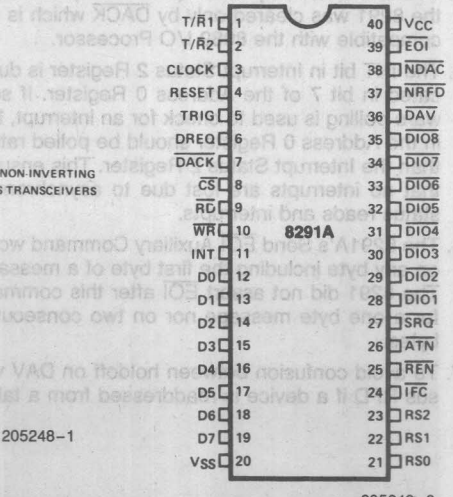


Figure 1. Block Diagram



205248-1

205248-2

Figure 2. Pin Configuration

November 1986
Order Number: 205248-002

8291A FEATURES AND IMPROVEMENTS

The 8291A is an improved design of the 8291 GPIB Talker/Listener. Most of the functions are identical to the 8291, and the pin configuration is unchanged.

The 8291A offers the following improvements to the 8291:

1. $\overline{EOI}$ is active with the data as a ninth data bit rather than as a control bit. This is to comply with some additions to the 1975 IEEE-488 Standard incorporated in the 1978 Standard.
2. The BO interrupt is not asserted until RFD is true. If the Controller asserts $\overline{ATN}$ synchronously, the data is guaranteed to be transmitted. If the Controller asserts $\overline{ATN}$ asynchronously, the SH (Source Handshake) will return to SIDS (Source Idle State), and the output data will be cleared. Then, if $\overline{ATN}$ is released while the 8291A is addressed to talk, a new BO interrupt will be generated. This change fixes 8291 problems which caused data to be lost or repeated and a problem with the RQS bit (sometimes cannot be asserted while talking).
3. LLOC and REMC interrupts are setting flipflops rather than toggling flipflops in the interrupt backup register. This ensures that the CPU knows that these state changes have occurred. The actual state can be determined by checking the LLO and REM status bits in the upper nibble of the Interrupt Status 2 Register.
4. DREQ is cleared by $\overline{DACK}$ ($\overline{RD}$ + $\overline{WR}$). DREQ on the 8291 was cleared only by $\overline{DACK}$ which is not compatible with the 8089 I/O Processor.
5. The INT bit in Interrupt Status 2 Register is duplicated in bit 7 of the Address 0 Register. If software polling is used to check for an interrupt, INT in the Address 0 Register should be polled rather than the Interrupt Status 2 Register. This ensures that no interrupts are lost due to asynchronous status reads and interrupts.
6. The 8291A's Send $\overline{EOI}$ Auxiliary Command works on any byte including the first byte of a message. The 8291 did not assert $\overline{EOI}$ after this command for a one byte message nor on two consecutive bytes.
7. To avoid confusion between holdoff on DAV versus RFD if a device is readdressed from a talker to a listener role or vice-versa during a holdoff, the "Holdoff on Source Handshake" has been eliminated. Only "Holdoff on Acceptor Handshake" is available.
8. The rsv local message is cleared automatically upon exit from SPAS if (APRS:STRS:SPAS) occurred. The automatic resetting of the bit after the serial poll is complete simplifies the service request software.
9. The SPASC interrupt on the 8291 has been replaced by the SPC (Serial Poll Complete) interrupt on the 8291A. SPC interrupt is set on exit from SPAS if APRS:STRS:SPAS occurred, indicating that the controller has read the bus status byte after the 8291A requested service. The SPASC interrupt was ambiguous because a controller could enter SPAS and exit SPAS generating two SPASC interrupts without reading the serial poll status byte. The SPC interrupt also simplifies the CPU's software by eliminating the interrupt when the serial poll is half way done.
10. The rtl Auxiliary Command in the 8291 has been replaced by Set and Clear rtl Commands in the 8291A. Using the new commands, the CPU has the flexibility to extend the length of local mode or leave it as a short pulse as in the 8291.
11. A holdoff RFD on GET, SDC, and DCL feature has been added to prevent additional bus activity while the CPU is responding to any of these commands. The feature is enabled by a new bit (B_4) in the Auxiliary Register B.
12. On the 8291, BO could cease to occur upon $\overline{IFC}$ going false if $\overline{IFC}$ occurred asynchronously. On the 8291A, BO continues to occur after $\overline{IFC}$ has gone false even if it arrived asynchronously.
13. User's software can distinguish between the 8291 and the 8291A as follows:
    a) pon (00H to register 5)
    b) RESET (02H to register 5)
    c) Read Interrupt Status 1 Register. If BO interrupt is set, the device is the 8291. If BO is clear, it is the 8291A.

    This can be used to set a flag in the user's software which will permit special routines to be executed for each device. It could be included as part of a normal initialization procedure as the first step after a chip reset.

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D_0–D_7	12–19	I/O	**DATA BUS PORT:** To be connected to microprocessor data bus.
RS_0–RS_2	21–23	I	**REGISTER SELECT:** Inputs, to be connected to three nonmultiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of $\overline{RD}$ ($\overline{WR}$).
$\overline{CS}$	8	I	**CHIP SELECT:** When low, enables reading from or writing into the register selected by RS_0–RS_2.
$\overline{RD}$	9	I	**READ STROBE:** When low with $\overline{CS}$ or $\overline{DACK}$ low, selected register contents are read.
$\overline{WR}$	10	I	**WRITE STROBE:** When low with $\overline{CS}$ or $\overline{DACK}$ low, data is written into the selected register.
INT ($\overline{INT}$)	11	O	**INTERRUPT REQUEST:** To the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU. May be software configured to be active low.
DREQ	6	O	**DMA REQUEST:** Normally low, set high to indicate byte output or byte input in DMA mode; reset by $\overline{DACK}$.
$\overline{DACK}$	7	I	**DMA ACKNOWLEDGE:** When low, resets DREQ and selects data in/data out register for DMA data transfer (actual transfer done by $\overline{RD}$/$\overline{WR}$ pulse). Must be high if DMA is not used.
TRIG	5	O	**TRIGGER OUTPUT:** Normally low; generates a triggering pulse with 1 μsec min. width in response to the GET bus command or Trigger auxiliary command.
CLOCK	3	I	**EXTERNAL CLOCK:** Input, used only for T, delay generator. May be any speed in 1–8 MHz range.
RESET	4	I	**RESET INPUT:** When high, forces the device into an "idle" (initialization) mode. The device will remain at "idle" until released by the microprocessor, with the "Immediate Execute pon" local message.
$\overline{DIO_1}$–$\overline{DIO_8}$	28–35	I/O	**8-BIT GPIB DATA PORT:** Used for bidirectional data byte transfer between 8291A and GPIB via non-inverting external line transceivers.
$\overline{DAV}$	36	I/O	**DATA VALID:** GPIB handshake control line. Indicates the availability and validity of information on the $\overline{DIO_1}$–$\overline{DIO_8}$ and $\overline{EOI}$ lines.
$\overline{NRFD}$	37	I/O	**NOT READY FOR DATA:** GPIB handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.
$\overline{NDAC}$	38	I/O	**NOT DATA ACCEPTED:** GPIB handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.
$\overline{ATN}$	26	I	**ATTENTION:** GPIB command line. Specifies how data on $\overline{DIO}$ lines are to be interpreted.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{IFC}$	24	I	**INTERFACE CLEAR:** GPIB command line. Places the interface functions in a known quiescent state.
$\overline{SRQ}$	27	O	**SERVICE REQUEST:** GPIB command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB.
$\overline{REN}$	25	I	**REMOTE ENABLE:** GPIB command line. Selects (in conjunction with other messages) remote or local control of the device.
$\overline{EOI}$	39	I/O	**END OR IDENTITY:** GPIB command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with $\overline{ATN}$, addresses the device during a polling sequence.
T/$\overline{R}$1	1	O	**EXTERNAL TRANSCEIVERS CONTROL LINE:** Set high to indicate output data/signals on the $\overline{DIO_1}$–$\overline{DIO_8}$ and $\overline{DAV}$ lines and input signals on the $\overline{NRFD}$ and $\overline{NDAC}$ lines (active source handshake). Set low to indicate input data/signals on the $\overline{DIO_1}$–$\overline{DIO_8}$ and $\overline{DAV}$ lines and output signals on the $\overline{NRFD}$ and $\overline{NDAC}$ lines (active acceptor handshake).
T/$\overline{R}$2	2	O	**EXTERNAL TRANSCEIVERS CONTROL LINE:** Set to indicate output signals on the $\overline{EOI}$ line. Set low to indicate expected input signal on the $\overline{EOI}$ line during parallel poll.
V_{CC}	40	P.S.	**POSITIVE POWER SUPPLY:** (5V ±10%).
GND	20	P.S.	**CIRCUIT GROUND POTENTIAL.**

NOTE:
All signals on the 8291A pins are specified with positive logic. However, IEEE 488 specifies negative logic on its 16 signal lines. Thus, the data is inverted once from D_0–D_7 to $\overline{DIO_0}$–$\overline{DIO_8}$ and non-inverting bus transceivers should be used.

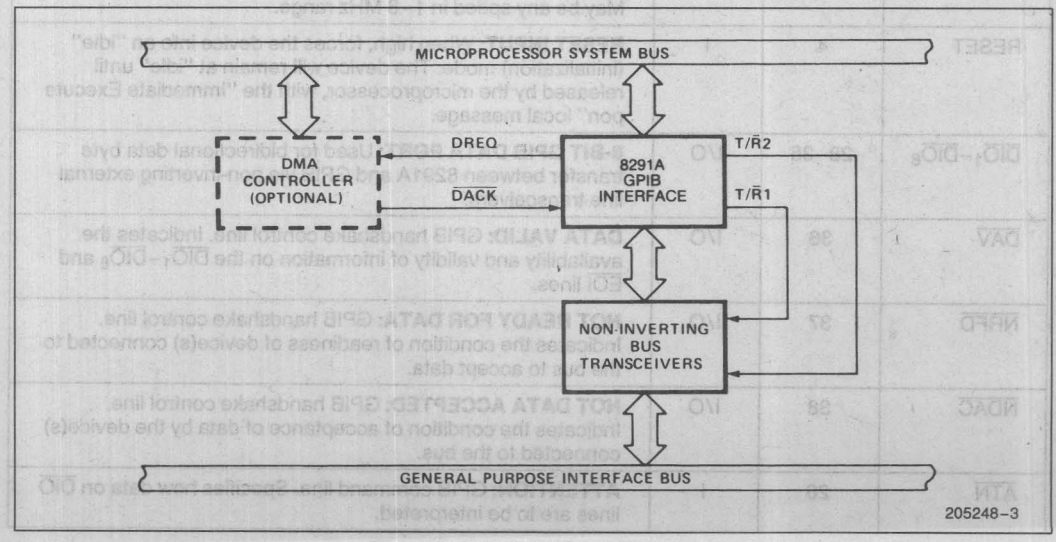


Figure 3. 8291A System Diagram

THE GENERAL PURPOSE INTERFACE BUS (GPIB)

The General Purpose Interface Bus (GPIB) is defined in the IEEE Standard 488-1978 "Digital Interface for Programmable Instrumentation." Although a knowledge of this standard is assumed, Figure 4 provides the bus structure for quick reference. Also, Tables 2 and 3 reference the interface state mnemonics and the interface messages respectively. Modified state diagrams for the 8291A are presented in Appendix A.

General Description

The 8291A is a microprocessor-controlled device designed to interface microprocessors, e.g., 8048/49, 8051, 8080/85, 8086/88 to the GPIB. It implements all of the interface functions defined in the IEEE-488 Standard except for the controller function. If an implementation of the Standard's Controller is desired, it can be connected with an Intel® 8292 to form a complete interface.

The 8291A handles communication between a microprocessor-controlled device and the GPIB. Its capabilities include data transfer, handshake protocol, talker/listener addressing procedures, device clearing and triggering, service request, and both serial and parallel polling. In most procedures, it does not disturb the microprocessor unless a byte has arrived (input buffer full) or has to be sent out (output buffer empty).

The 8291A architecture includes 16 registers. Eight of these registers may be written into by the microprocessor. The other eight registers may be read by the microprocessor. One each of these read and write registers is for direct data transfers. The rest of the write registers control the various features of the chip, while the rest of the read registers provide the microprocessor with a monitor of GPIB states, various bus conditions, and device conditions.

GPIB Addressing

Each device connected to the GPIB must have at least one address whereby the controller device in charge of the bus can configure it to talk, listen, or send status. An 8291A implementation of the GPIB offers the user three alternative addressing modes for which the device can be initialized for each application. The first of these modes allows for the device to have two separate primary addresses. The second mode allows the user to implement a single talker/listener with a two byte address (primary address + secondary address). The third mode again allows for two distinct addresses but in this instance, they can each have a ten-bit address (5 low-order bits of each of two bytes). However, this mode requires that the secondary addresses be passed to the microprocessor for verification. These three addressing schemes are described in more detail in the discussion of the Address Registers.

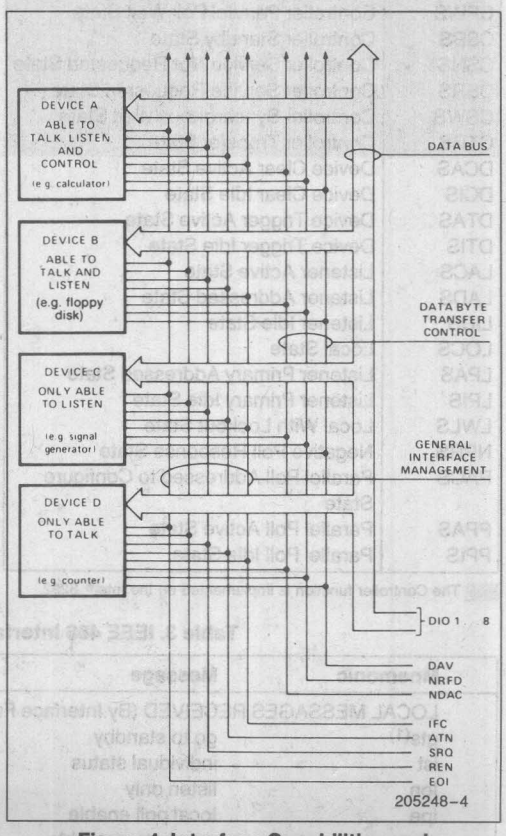


Figure 4. Interface Capabilities and Bus Structure

Table 2. IEEE 488 Interface State Mnemonics

Mnemonic	State Represented
ACDS	Accept Data State
ACRS	Acceptor Ready State
AIDS	Acceptor Idle State
ANRS	Acceptor Not Ready State
APRS	Affirmative Poll Response State
AWNS	Acceptor Wait for New Cycle State
CACS	Controller Active State
CADS	Controller Addressed State
CAWS	Controller Active Wait State
CIDS	Controller Idle State
CPPS	Controller Parallell Poll State
CPWS	Controller Parallel Poll Wait State
CSBS	Controller Standby State
CSNS	Controller Service Not Requested State
CSRS	Controller Service Requested State
CSWS	Controller Synchronous Wait State
CTRS	Controller Transfer State
DCAS	Device Clear Active State
DCIS	Device Clear Idle State
DTAS	Device Trigger Active State
DTIS	Device Trigger Idle State
LACS	Listener Active State
LADS	Listener Addressed State
LIDS	Listener Idle State
LOCS	Local State
LPAS	Listener Primary Addressed State
LPIS	Listener Primary Idle State
LWLS	Local With Lockout State
NPRS	Negative Poll Response State
PACS	Parallel Poll Addressed to Configure State
PPAS	Parallel Poll Active State
PPIS	Parallel Poll Idle State
PPSS	Parallel Poll Standby State
PUCS	Parallel Poll Unaddressed to Configure State
REMS	Remote State
RWLS	Remote With Lockout State
SACS	System Control Active State
SDYS	Source Delay State
SGNS	Source Generate State
SIAS	System Control Interface Clear Active State
SIDS	Source Idle State
SIIS	System Control Interface Clear Idle State
SINS	System Control Interface Clear Not Active State
SIWS	Source Idle Wait State
SNAS	System Control Not Active State
SPAS	Serial Poll Active State
SPIS	Serial Poll Idle State
SPMS	Serial Poll Mode State
SRAS	System Control Remote Enable Active State
SRIS	System Control Remote Enable Idle State
SRNS	System Control Remote Enable Not Active State
SRQS	Service Request State
STRS	Source Transfer State
SWNS	Source Wait for New Cycle State
TACS	Talker Active State
TADS	Talker Addressed State
TIDS	Talker Idle State
TPIS	Talker Primary Idle State

The Controller function is implemented on the Intel® 8292.

Table 3. IEEE 488 Interface Message Reference List

Mnemonic	Message	Interface Function(s)
LOCAL MESSAGES RECEIVED (By Interface Functions)		
gts(1)	go to standby	C
ist	individual status	PP
lon	listen only	L, LE
lpe	local poll enable	PP
nba	new byte available	SH
pon	power on	SH, AH, T, TE, L, LE, SR, RL, PP, C
rdy	ready	AH
rpp(1)	request parallel poll	C
rsc(1)	request system control	C
rsv	request service	SR
rtl	return to local	RL
sic(1)	send interface clear	C
sre(1)	send remote enable	C
tca(1)	take control asynchronously	C

Table 3. IEEE 488 Interface Message Reference List (Continued)

Mnemonic	Message	Interface Function(s)
tcs(1)	take control synchronously	AH, C
ton	talk only	T, TE
REMOTE MESSAGES RECEIVED		
ATN	Attention	SH, AH, T, TE, L, LE, PP, C
DAB	Data Byte	(Via L, LE)
DAC	Data Accepted	SH
DAV	Data Valid	AH
DCL	Device Clear	DC
END	End	(via L, LE)
GET	Group Execute Trigger	DT
GTL	Go to Local	RL
IDY	Identify	L, LE, PP
IFC	Interface Clear	T, TE, L, LE, C
LLO	Local Lockout	RL
MLA	My Listen Address	L, LE, RL, T, TE
MSA	My Secondary Address	TE, LE, RL
MTA	My Talk Address	T, TE, L, LE
OSA	Other Secondary Address	TE
OTA	Other Talk Address	T, TE
PCG	Primary Command Group	TE, LE, PP
PPC(2)	Parallel Poll Configure	PP
[PPD](2)	Parallel Poll Disable	PP
[PPE](2)	Parallel Poll Enable	PP
PPR_N(1)	Parallel Poll Response N	(via C)
PPU(2)	Parallel Poll Unconfigure	PP
REN	Remote Enable	RL
RFD	Ready for Data	SH
RQS	Request Service	(via L, LE)
[SDC]	Select Device Clear	DC
SPD	Serial Poll Disable	T, TE
SPE	Serial Poll Enable	T, TE
SQR(1)	Service Request	(via C)
STB	Status Byte	(via L, LE)
TCT or [TCT](1)	Take Control	C
UNL	Unlisten	L, LE
REMOTE MESSAGES SENT		
ATN	Attentions	C
DAB	Data Byte	(Via T,TE)
DAC	Data Accepted	AH
DAV	Data Valid	SH
DCL	Device Clear	(via C)
END	End	(via T)
GET	Group Execute Trigger	(via C)
GTL	Go to Local	(via C)
IDY	Identify	C
IFC	Interface Clear	C
LLO	Local Lockout	(via C)
MLA or [MLA]	My Listen Address	(via C)
MSA or [MSA]	My Secondary Address	(via C)
MTA or [MTA]	My Talk Address	(via C)
OSA	Other Secondary Address	(via C)

Table 3. IEEE 488 Interface Message Reference List (Continued)

Mnemonic	Message	Interface Function(s)(3)
OTA	Other Talk Address	(via C)
PCG	Primary Command Group	(via C)
PPC	Parallel Poll Configure	(via C)
[PPD]	Parallel Poll Disable	(via C)
[PPE]	Parallel Poll Enable	(via C)
PPR_N	Parallel Poll Response N	PP
PPU	Parallel Poll Unconfigure	(via C)
REN	Remote Enable	C
RFD	Ready for Data	AH
RQS	Request Service	T, TE
[SDC]	Selected Device Clear	(via C)
SPD	Serial Poll Disable	(via C)
SPE	Serial Poll Enable	(via C)
SRQ	Service Request	SR
STB	Status Byte	(via T,TE)
TCT	Take Control	(via C)
UNL	Unlisten	(via C)

NOTES:
1. These messages are handled only by Intel's 8292.
2. Undefined commands which may be passed to the microprocessor.
3. All Controller messages must be sent via Intel's 8292.

8291A Registers

A bit-by-bit map of the 16 registers on the 8291A is presented in Figure 5. A more detailed explanation of each of these registers and their functions follows. The access of these registers by the microprocessor is accomplished by using the $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, and RS_0–RS_2 pins.

Register	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	RS_0–RS_2
All Read Registers	0	0	1	CCC
All Write Registers	0	1	0	CCC
High Impedance	1	d	d	ddd

Data Registers

D17	D16	D15	D14	D13	D12	D11	D10

DATA-IN REGISTER (0R)

DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

DATA-OUT REGISTER (0W)

The Data-In Register is used to move data from the GPIB to the microprocessor or to memory when the 8291A is addressed to listen. Incoming information is separately latched by this register, and its contents are not destroyed by a write to the data-out register. The RFD (Ready for Data) message is held false until the byte is removed from the data in register, either by the microprocessor or by DMA. The 8291A then completes the handshake automatically. In RFD holdoff mode (see Auxiliary Register A), the handshake is not finished until a command is sent telling the 8291A to release the holdoff. In this way, the same byte may be read several times, or an over anxious talker may be held off until all available data has been processed.

When the 8291A is addressed to talk, it uses the data-out register to move data onto the GPIB. After the BO interrupt is received and a byte is written to this register, the 8291A initiates and completes the handshake while sending the byte out over the bus. In the BO interrupt disable mode, the user should wait until BO is active before writing to the register. (In the DMA mode, this will happen automatically.) A read of the Data-In Register does not destroy the information in the Data-Out Register.

Interrupt Registers

CPT	APT	GET	END	DEC	ERR	BO	BI

INTERRUPT STATUS 1 (1R)

INT	SPAS	LLO	REM	SPC	LLOC	REMC	ADSC

INTERRUPT STATUS 2 (2R)

CPT	APT	GET	END	DEC	ERR	BO	BI

INTERRUPT ENABLE 1 (1W)

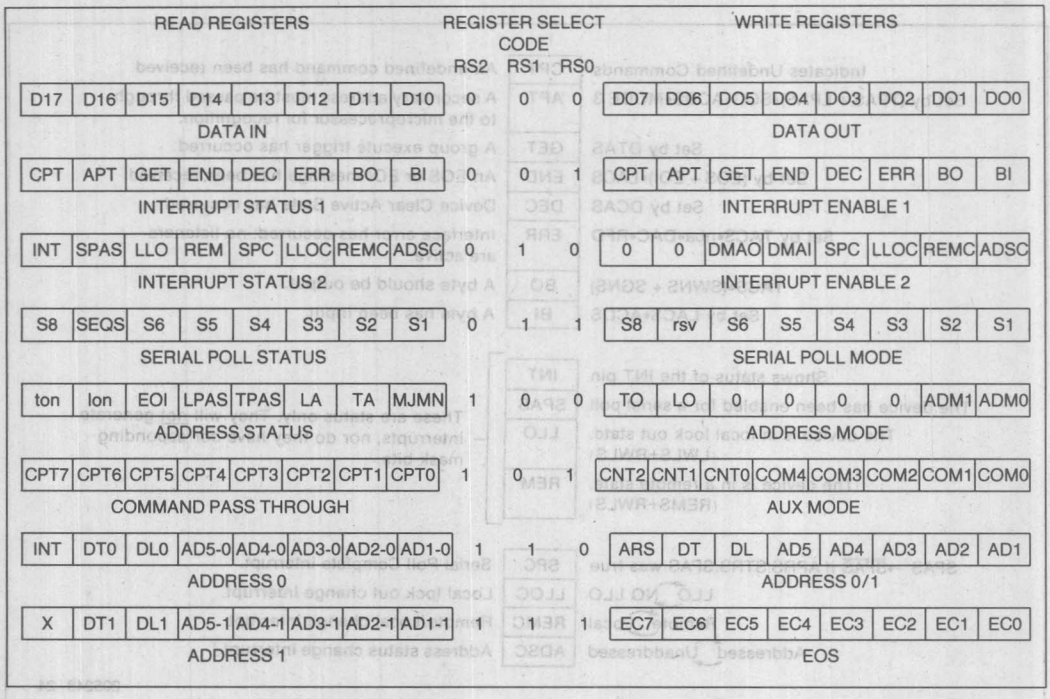
0	0	DMAO	DMAI	SPC	LLOC	REMC	ADSC

INTERRUPT ENABLE 2 (2W)

INT	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0

ADDRESS 0 REGISTER

Figure 5. 8291A Registers



The 8291A can be configured to generate an interrupt to the microprocessor upon the occurrence of any of 12 conditions or events on the GPIB. Upon receipt of an interrupt, the microprocessor must read the Interrupt Status Registers to determine which event has occurred, and then execute the appropriate service routine (if necessary). Each of the 12 interrupt status bits has a matching enable bit in the interrupt enable registers. These enable bits are used to select the events that will cause the INT pin to be asserted. Writing a logic "1" into any of these bits enables the corresponding interrupt status bits to generate an interrupt. Bits in the Interrupt Status Registers are set regardless of the states of the enable bits. The Interrupt Status Registers are then cleared upon being read or when a local pon (power-on) message is executed. If an event occurs while one of the Interrupt Status Registers is being read, the event is held until after its register is cleared and then placed in the register.

NOTE:
Reading the interrupt status registers clears the bits which were set. The software must examine *all* relevant bits in the interrupt status registers before disregarding the value or an important interrupt may be missed.

The mnemonics for each of the bits in these registers and a brief description of their respective functions appears in Table 4. This table also indicates how each of the interrupt bits is set.

NOTE:
The INT bit in the Address 0 Register is a duplicate of the INT bit in the Interrupt Status 2 Register. It is only a status bit. It does not generate interrupts and thus does not have a corresponding enable bit.

The BO and BI interrupts enable the user to perform data transfer cycles. BO indicates that a data byte should be written to the Data Out Register. It is set by TACS • (SWNS + SGNS) • RFD. It is reset when the data byte is written, ATN is asserted, or the 8291A exits TACS. Data should never be written to the Data Out Register before BO is set. Similarly, BI is set when an input byte is accepted into the 8291A and reset when the microprocessor reads the Data In Register. BO and BI are also reset by pon (power-on local message) and by a read of the Interrupt Status 1 Register. However, if it is so desired, data transfer cycles may be performed without reading the Interrupt Status 1 Register if all interrupts except for BO or BI are disabled; BO and BI will automatically reset after each byte is transferred.

Table 4. Interrupt Bits

Indicates Undefined Commands	CPT	An undefined command has been received.
Set by (TPAS + LPAS)•SCG•ACDS•MODE 3	APT	A secondary address must be passed through to the microprocessor for recognition.
Set by DTAS	GET	A group execute trigger has occurred.
Set by (EOS + EOI)•LACS	END	An EOS or EOI message has been received.
Set by DCAS	DEC	Device Clear Active State has occurred.
Set by TACS•nba•DAC•RFD	ERR	Interface error has occurred; no listeners are active.
TACS•(SWNS + SGNS)	BO	A byte should be output.
Set by LACS•ACDS	BI	A byte has been input.
Shows status of the INT pin	INT	These are status only. They will not generate interrupts, nor do they have corresponding mask bits.
The device has been enabled for a serial poll	SPAS	
The device is in local lock out state. (LWLS+RWLS)	LLO	
The device is in a remote state. (REMS+RWLS)	REM	
SPAS →$\overline{\text{SPAS}}$ if APRS:STRS:SPAS was true	SPC	Serial Poll Complete interrupt.
LLO ⇄ NO LLO	LLOC	Local lock out change interrupt.
Remote ⇄ Local	REMC	Remote/Local change interrupt.
Addressed ⇄ Unaddressed	ADSC	Address status change interrupt.[1]

205248-24

NOTE:
1. In ton (talk-only) and lon (listen-only) modes, no ADSC interrupt is generated.

If the 8291A is used in the interrupt mode, the INT and DREQ pins can be dedicated to data input and output interrupts respectively by enabling BI and DMAO, provided that no other interrupts are enabled. This eliminates the need to read the interrupt status registers if a byte is received or transmitted.

The ERR bit is set to indicate the bus error condition when the 8291A is an active talker and tries to send a byte to the GPIB, but there are no active listeners (e.g., all devices on the GPIB are in AIDS). The logical equivalent of (nba • TACS • DAC • RFD) will set this bit.

The DEC bit is set whenever DCAS has occurred. The user must define a known state to which all device functions will return in DCAS. Typically this state will be a power-on state. However, the state of the device functions at DCAS is at the designer's discretion. It should be noted that DCAS has no effect on the interface functions which are returned to a known state by the IFC (interface clear) message or the pon local message.

The END interrupt bit may be used by the microprocessor to detect that a multi-byte transfer has been completed. The bit will be set when the 8291A is an active listener (LACS) and either EOS (provided the End on EOS Received feature is enabled in the Auxiliary Register A) or EOI is received. EOS will generate an interrupt when the byte in the Data In Register matches the byte in the EOS register. Otherwise the interrupt will be generated when a true input is detected on EOI.

The GET interrupt bit is used by the microprocessor to detect that DTAS has occurred. It is set by the 8291A when the GET message is received while it is addressed to listen. The TRIG output pin of the 8291A fires when the GET message is received. Thus, the basic operation of device trigger may be started without microprocessor software intervention.

The APT interrupt bit indicates to the processor that a secondary address is available in the CPT register for validation. This interrupt will only occur if Mode 3 addressing is in effect. (Refer to the section on addressing.) In Mode 2, secondary addresses will be recognized automatically on the 8291A. They will be ignored in Mode 1.

The CPT interrupt bit flags the occurrence of an undefined command and of all secondary commands following an undefined command. The Command Pass Through feature is enabled by the B0 bit of Auxiliary Register B. Any message not decoded by the 8291A (not included in the state diagrams in Appendix B) becomes an undefined command. Note that any addressed command is automatically ignored when the 8291A is not addressed.

Undefined commands are read by the CPU from the Command Pass Through register of the 8291A. This register reflects the logic levels present on the data lines at the time it is read. If the CPT feature is enabled, the 8291A will hold off the handshake until this register is read.

An especially useful feature of the 8291A is its ability to generate interrupts from state transitions in the interface functions. In particular, the lower 3 bits of the Interrupt Status 2 Register, if enabled by the corresponding enable bits, will cause an interrupt upon changes in the following states as defined in the IEEE 488 Standard.

Bit 0	ADSC	change in LIDS or TIDS or MJMN
Bit 1	REMC	change in LOCS or REMS
Bit 2	LLOC	change in LWLS or RWLS

The upper 4 bits of the Interrupt Status 2 Register are available to the processor as status bits. Thus, if one of the bits 0–2 generates an interrupt indicating a state change has taken place, the corresponding status bit (bits 3–5) may be read to determine what the new state is. To determine the nature of a change in addressed status (bit 0) the Address Status Register is available to be read. The SPC interrupt (bit 3 in Interrupt Status 2) is set upon exit from SPAS if APRS:STRS:SPAS occurred which indicates that the GPIB controller has read the bus serial poll status byte after the 8291A requested service (asserted SRQ). The SPC interrupt occurs once after the controller reads the status byte if service was requested. The controller may read the status byte later, and the byte will contain the last status the 8291A's CPU wrote to the Serial Poll Mode Register, but the SRQS bit will not be set and no interrupt will be generated. Finally, bit 7 monitors the state of the 8291A INT pin. Logically, it is an OR of all enabled interrupt status bits. One should note that bits 3–6 of the Interrupt Status 2 Register do not generate interrupts, but are available only to be read as status bits by the processor. Bit 7 in Interrupt Status 2 is duplicated in Address 0 Register, and the latter should be used when polling for interrupts to avoid losing one of the interrupts in Interrupt Status 2 Register.

Bits 4 and 5 (DMAI, DMAO) of the Interrupt Mask 2 Register are available to enable direct data transfers between memory and the GPIB; DMAI (DMA in) enables the DREQ (DMA request) pin of the 8291A to be asserted upon the occurrence of BI. Similarly, DMAO (DMA out) enables the DREQ pin to be asserted upon the occurrence of BO. One might note that the DREQ pin may be used as a second interrupt output pin, monitoring BI and/or BO and enabled by DMAI and DMAO. One should note that the DREQ pin is not affected by a read of the Interrupt Status 1 Register. It is reset whenever a byte is written to the Data Out Register or read from the Data In Register.

To ensure that an interrupt status bit will not be cleared without being read, and will not remain uncleared after being read, the 8291A implements a special interrupt handling procedure. When an enabled interrupt bit is set in either of the Interrupt Status Registers, the input of the registers are blocked until the set bit is read and reset by the microprocessor. Thus, potential problems arise when interrupt status changes while the register is being blocked. However, the 8291A stores all new interrupts in a temporary register and transfers them to the appropriate Interrupt Status Register after the interrupt has been reset. This transfer takes place only if the corresponding bits were read as zeroes.

3

Serial Poll Registers

S8	SRQS	S6	S5	S4	S3	S2	S1

SERIAL POLL STATUS (3R)

S8	rsv	S6	S5	S4	S3	S2	S1

SERIAL POLL MODE (3W)

The Serial Poll Mode Register determines the status byte that the 8291A sends out on the GPIB data lines when it receives the SPE (Serial Poll Enable) message. Bit 6 of this register is reserved for the rsv (request service) local message. Setting this bit to 1 causes the 8291A to assert its $\overline{SRQ}$ line, indicating its need for attention from the controller-in-charge of the GPIB. The other bits of this register are available for sending status information over the GPIB. Sometime after the microprocessor initiates a request for service by setting bit 6, the controller of the GPIB sends the SPE message and then addresses the 8291A to talk. At this point, one byte of status is returned by the 8291A via the Serial Poll Mode Register. After the status byte is read by the controller, rsv is automatically cleared by the 8291A and an SPC interrupt is generated. The CPU may request service again by writing another byte to the Serial Poll Mode Register with the rsv bit set. If the control-

ler performs a serial poll when the rsv bit is clear, the last status byte written will be read, but the $\overline{SRQ}$ line will not be driven by the 8291A and the SRQS bit will be clear in the status byte.

The Serial Poll Status Register is available for reading the status byte in the Serial Poll Mode Register. The processor may check the status of a request for service by polling bit 6 of this register, which corresponds to SRQS (Service Request State). When a Serial Poll is conducted and the controller-in-charge reads the status byte, the SRQS bit is cleared. The $\overline{SRQ}$ line and the rsv bit are tied together.

Address Registers

ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN

ADDRESS STATUS (4R)

INT	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0

ADDRESS 0 (6R)

X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1

ADDRESS 1 (7R)

TO	LO	0	0	0	0	ADM1	ADM0

ADDRESS MODE (4W)

ARS	DT	DL	AD5	AD4	AD3	AD2	AD1

ADDRESS 0/1 (6W)

The Address Mode Register is used to select one of the five modes of addressing available on the 8291A. It determines the way in which the 8291A uses the information in the Address 0 and Address 1 Registers.

—In Mode 1, the contents of the Address 0 Register constitute the "Major" talker/listener address while the Address 1 Register represents the "Minor" talker/listener address. In applications where only one address is needed, the major talker/listener is used, and the minor talker/listener should be disabled. Loading an address via the Address 0/1 Register into Address Registers 0 and 1 enables the major and minor talker/listener functions respectively.

—In Mode 2 the 8291A recognizes two sequential address bytes: a primary followed by a secondary. Both address bytes must be received in order to enable the device to talk or listen. In this manner, Mode 2 addressing implements the extended talker and listener functions as defined in IEEE-488.

To use Mode 2 addressing the primary address must be loaded into the Address 0 Register, and the Secondary Address is placed in the Address 1 Register. With both primary and secondary addresses residing on chip, the 8291A can handle all addressing sequences without processor intervention.

—In Mode 3, the 8291A handles addressing just as it does in Mode 1, except that each Major or Minor primary address must be followed by a secondary address. All secondary addresses must be verified by the microprocessor when Mode 3 is used. When the 8291A is in TPAS or LPAS (talker/listener primary addressed state), and it does not recognize the byte on the DIO lines, an APT interrupt is generated (see section on Interrupt Registers) and the byte is available in the CPT (Command Pass-Through) Register. As part of its interrupt service routine, the microprocessor must read the CPT Register and write one of the following responses to the Auxiliary Mode Register:

1. 07H implies a non-valid secondary address
2. 0FH implies a valid secondary address

Setting the TO bit generates the local ton (talk-only) message and sets the 8291A to a talk-only mode. This mode allows the device to operate as a talker in an interface system without a controller.

Setting the LO bit generates the local lon (listen-only) message and sets the 8291A to a listen-only mode. This mode allows the device to operate as a listener in an interface system without a controller. The above bits may also be used by a controller-in-charge to set itself up for remote command or data communication.

The mode of addressing implemented by the 8291A may be selected by writing one of the following bytes to the Address Mode Register.

Register Contents	Mode
10000000	Enable talk only mode (ton)
01000000	Enable listen only mode (lon)
11000000	The 8291 may talk to itself
00000001	Mode 1, (Primary-Primary)
00000010	Mode 2 (Primary-Secondary)
00000011	Mode 3 (Primary/APT-Primary/APT)

The Address Status Register contains information used by the microprocessor to handle its own addressing. This information includes status bits that monitor the address state of each talker/listener, "ton" and "lon" flags which indicate the talk and listen only states, and an EOI bit which, when set, signifies that the END message came with the last data byte. LPAS and TPAS indicate that the listener

or talker primary address has been received. The microprocessor can use these bits when the secondary address is passed through to determine whether the 8291A is addressed to talk or listen. The LA (listener addressed) bit will be set when the 8219A is in LACS (Listener Active State) or in LADS (Listener Addressed State). Similarly, the TA (Talker Addressed bit) will be set to indicate TACS or TADS, but also to indicate SPAS (Serial Poll Active State). The MJMN bit is used to determine whether the information in the other bits applies to the Major or Minor talker/listener. It is set to "1" when the Minor talker/listener is addressed. It should be noted that only one talker/listener may be active at any one time. Thus, the MJMN bit will indicate which, if either, of the talker/listeners is addressed or active.

The Address 0/1 Register is used for specifying the device's addresses according to the format selected in the Address Mode Register. Five bit addresses may be loaded into the Address 0 and Address 1 Registers by writing into the Address 0/1 Register. The ARS bit is used to select which of these registers the other seven bits will be loaded into. The DT and DL bits may be used to disable the talker or listener function at the address signified by the other five bits. When Mode 1 addressing is used and only one primary address is desired, *both* the talker and the listener should be disabled at the Minor address.

As an example of how the Address 0/1 Register might be used, consider an example where two primary addresses are needed in the device. The Major primary address will be selectable only as a talker and the Minor primary address will be selectable only as a listener. This configuration of the 8291A is formed by the following sequence of writes by the microprocessor.

Operation	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	Data	RS_2–RS_0
1. Select addressing Mode 1	0	1	0	00000001	100
2. Load major address into Address 0 Register with listener function disabled.	0	1	0	001AAAAA	110
3. Load minor address into Address 1 Register with talker function disabled.	0	1	0	110BBBBB	110

At this point, the addresses AAAAA and BBBBB are stored in the Address 0 and Address 1 Registers respectively, and are available to be read by the microprocessor. Thus, it is not necessary to store any address information elsewhere. Also, with the information stored in the Address 0 and Address 1 Registers, processor intervention is not required to recognize addressing by the controller. Only in Mode 3, where secondary addresses are passed through, must the processor intervene in the addressing sequence.

The Address 0 Register contains a copy of bit 7 of the Interrupt Status 2 Register (INT). This is to be used when polling for interrupts. Software should poll register 6 checking for INT (bit 7) to be set. When INT is set, the Interrupt Status Register should be read to determine which interrupt was received.

Command Pass Through Register

CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0

COMMAND PASS THROUGH (5R)

The Command Pass Through Register is used to transfer undefined 8-bit remote message codes from the GPIB to the microprocessor. When the CPT feature is enabled (bit BO in Auxiliary Register B), any message not decoded by the 8291A becomes an undefined command. When Mode 3 addressing is used secondary addresses are also passed through the CPT Register. In either case, the 8291A will hold-off the handshake until the microprocessor reads this register and issues the VSCMD auxiliary command.

The CPT and APT interrupts flag the availability of undefined commands and secondary addresses in the CPT Register. The details of these interrupts are explained in the section on Interrupt Registers.

An added feature of the 8291A is its ability to handle undefined secondary commands following undefined primaries. Thus, the number of available commands for future IEEE-488 definition is increased; one undefined primary command followed by a sequence of as many as 32 secondary commands can be processed. The IEEE-488 Standard does not permit users to define their own commands, but upgrades of the standard are thus provided for.

The recommended use of the 8291A's undefined command capabilities is for a controller-configured Parallel Poll. The PPC message is an undefined primary command typically followed by PPE, and undefined secondary command. For details on this procedure, refer to the section on Parallel Poll Protocol.

Auxiliary Mode Register

CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0

AUX MODE (5W)
CNT0–2:CONTROL BITS
COM0–4:COMMAND BITS

The Auxiliary Mode Register contains a three-bit control field and a five-bit command field. It is used for several purposes on the 8291A:

1. To load "hidden" auxiliary registers on the 8291A.
2. To issue commands from the microprocessor to the 8291A.
3. To preset an internal counter used to generate T1, delay in the Source Handshake function, as defined in IEEE-488.

Table 5 summarizes how these tasks are performed with the Auxiliary Mode Register. Note that the three control bits determine how the five command bits are interpreted.

Table 5

Code		Command
Control Bits	**Command Bits**	
000	0CCCC	Execute auxiliary command CCCC
001	ODDDD	Preset internal counter to match external clock frequency of DDDD MHz (DDDD binary representation of 1 to 8 MHz)
100	DDDDD	Write DDDDD into auxiliary register A
101	DDDDD	Write DDDDD into auxiliary register B
011	$USP_3P_2P_1$	Enable/disable parallel poll either in response to remote messages (PPC followed by PPE or PPD) or as a local lpe message. (Enable if U = 0, disable if U = 1.)

AUXILIARY COMMANDS

Auxiliary commands are executed by the 8291A whenever 0000CCCC is written into the Auxiliary Mode Register, where CCCC is the 4-bit command code.

0000—Immediate Execute pon: This command resets the 8291A to a power up state (local pon message as defined in IEEE-488).

The following conditions constitute the power up state:

1. All talkers and listeners are disabled.
2. No interrupt status bits are set.

The 8291A is designed to power up in certain states as specified in the IEEE-488 state diagrams. Thus, the following states are in effect in the power up state: SIDS, AIDS, TIDS, LIDS, NPRS, LOCS, and PPIS.

The "0000" pon is an immediate execute command (a pon pulse). It is also used to release the "initialize" state generated by either an external reset pulse or the "0010" Chip Reset command.

0010—Chip Reset (Initialize): This command has the same effect as a pulse applied to the Reset pin. (Refer to the section on Reset Procedure.)

0011—Finish Handshake: This command finishes a handshake that was stopped because of a holdoff on RFD. (Refer to Auxiliary Register A.)

0100—Trigger: A "Group Execute Trigger" is forced by this command. It has the same effect as a GET command issued by the controller-in-charge of the GPIB, but does not cause a GET interrupt.

0101, 1101—Clear/Set rtl: These commands correspond to the local rtl message as defined by the IEEE-488. The 8291A will go into local mode when a Set rtl Auxiliary Command is received if local lockout is not in effect. The 8291A will exit local mode after receiving a Clear rtl Auxiliary Command if the 8291A is addressed to listen.

0110—Send EOI: The EOI line of the 8291A may be asserted with this command. The command causes EOI to go true with the next byte transmitted. The EOI line is then cleared upon completion of the handshake for that byte.

0111, 1111—Non Valid/Valid Secondary Address or Command (VSCMD): This command informs the 8291A that the secondary address received by the microprocessor was valid or invalid (0111 = invalid, 1111 = valid). If Mode 3 addressing is used, the processor must field each extended address and respond to it, or the GPIB will hang up. Note that the COM3 bit is the invalid/valid flag.

The valid (1111) command is also used to tell the 8291A to continue from the command-pass-through-state, or from RFD holdoff on GET, SDC or DCL.

1000—pon: This command puts the 8291A into the pon (power on) state and holds it there. It is similar to a Chip Reset except none of the Auxiliary Mode Registers are cleared. In this state, the 8291A does not participate in any bus activity. An Immediate Execute pon releases the 8291A from the pon state and permits the device to participate in the bus activity again.

0001, 1001—Parallel Poll Flag (local "ist" message): This command sets (1001) or clears (0001) the parallel poll flag. A "1" is sent over the assigned data line (PRR = Parallel Poll Response true) only if the parallel poll flag matches the sense bit from the lpe local message (or indirectly from the PPE message). For a more complete description of the Parallel Poll features and procedures refer to the section on Parallel Poll Protocol.

INTERNAL COUNTER

The internal counter determines the delay time allowed for the setting of data on the DIO lines. This delay time is defined as T_1 in IEEE-488 and appears in the Source Handshake state diagram between the SDYS and STRS. As such, DAV is asserted T_1 after the DIO lines are driven. Consequently , T_1 is a major factor in determining the data transfer rate of the 8291A over the GPIB (T_1 = TWRDV2-TWRD15).

When open-collector transceivers are used for connection to the GPIB, T_1 is defined by IEEE-488 to be 2 μs. By writing 0010DDDD into the Auxiliary Mode Register, the counter is preset to match a f_C MHz clock input, where DDDD is the binary representation of N_F [$1 \leq N_F \leq 8$, $N_F = (DDDD)_2$]. When $N_F = f_C$, a 2 μs T_1 delay will be generated before each DAV asserted.

$$T_{1(\mu s)} = \frac{2N_F}{f_c} + t_{SYNC}, 1 \leq N_F \leq 8$$

t_{SYNC} is a synchronization error, greater than zero and smaller than the larger of T clock high and T clock low. (For a 50% duty cycle clock, t_{SYNC} is less than half the clock cycle).

If it is necessary that T_1 be different from 2 μs, N_F may be set to a value other than f_C. In this manner, data transfer rates may be programmed for a given system. In small systems, for example, where transfer rates exceeding GPIB specifications are required, one may set $N_F < f_C$ and decrease T_1.

When tri-state transceivers are used, IEEE-488 allows a higher transfer rate (lower T_1). Use of the 8291A with such transceivers is enabled by setting B_2 in Auxilliary Register B. In this case, setting $N_F = f_c$ causes a T_1 delay of 2μs to be generated for the first byte transmitted—all subsequent bytes will have a delay of 500 ns.

$$T_1 \text{ (High Speed) } \mu s = \frac{N_F}{2f_C} + t_{SYNC}$$

Thus, the shortest T_1 is achieved by setting $N_F = 1$ using an 8 MHz clock with a 50% duty cycle clock ($t_{SYNC} < 63$ ns):

$$T_{1(HS)} = \frac{1}{2 \times 8} + 0.063 = 125 \text{ ns max.}$$

AUXILIARY REGISTER A

Auxiliary Register A is a "hidden" 5-bit register which is used to enable some of the 8291A features. Whenever a 100 $A_4A_3A_2A_1A_0$ byte is written into the Auxiliary Register, it is loaded with the data $A_4A_3A_2A_1A_0$. Setting the respective bits to "1" enables the following features.

A_0—RFD Holdoff on all Data: If the 8291A is listening, RFD will not be sent true until the "finish handshake" auxiliary command is issued by the microprocessor. The holdoff will be in effect for each data byte.

A_1—RFD Holdoff on End: This feature enables the holdoff on EOI or EOS (if enabled). However, no hold-off will be in effect on any other data bytes.

A_2—End on EOS Received: Whenever the byte in the Data In Register matches the byte in the EOS Register, the END interrupt bit will be set in the Interrupt Status 1 Register.

A_3—Output EOI on EOS Sent: Any occurrence of data in the Data Out Register matching the EOS Register causes the EOI line to be sent true along with the data.

A_4—EOS Binary Compare: Setting this bit causes the EOS Register to function as a full 8-bit word. When it is not set, the EOS Register is a 7-bit word (for ASCII characters).

If $A_0 = A_1 = 1$, a special "continuous Acceptor Handshake cycling" mode is enabled. This mode should be used only in a controller system configuration, where both the 8291A and the 8292 are used. It provides a continuous cycling through the Acceptor Handshake state diagram, requiring no local messages from the microprocessor; the rdy local message is automatically generated when in ANRS. As such, the 8291A Acceptor Handshake serves as the controller Acceptor Handshake. Thus, the controller cycles through the Acceptor Handshake without delaying the data transfer in progress. When the tcs local message is executed, the 8291A should be taken out of the "continuous AH cycling" mode, the GPIB will hang up in ANRS, and a BI interrupt will be generated to indicate that control may be taken. A

simpler procedure may be used when a "tcs on end of block" is executed; the 8291A may stay in "continuous AH cycling". Upon the end of a block (EOI or EOS received), a holdoff is generated, the GPIB hangs up in ANRS, and control may be taken.

AUXILIARY REGISTER B

Auxiliary Register B is a "hidden" 4-bit register which is used to enable some of the features of the 8291A. Whenever a 101 $B_4B_3B_2B_1B_0$ is written into the Auxiliary Mode Register, it is loaded with the data $B_4B_3B_2B_1B_0$. Setting the respective bits to "1" enables the following features:

B_0—Enable Undefined Command Pass Through: This feature allows any commands not recognized by the 8291A to be handled in software. If enabled, this feature will cause the 8291A to holdoff the handshake when an undefined command is received. The microprocessor must then read the command from the Command Pass Through Register and send the VSCMD auxiliary command. Until the VSCMD command is sent, the handshake holdoff will be in effect.

B_1—Send EOI in SPAS: This bit enables EOI to be sent with the status byte; EOI is sent true in Serial Poll Active State. Otherwise, EOI is sent false in SPAS.

B_2—Enable High Speed Data Transfer: This feature may be enabled when tri-state external transceivers are used. The data transfer rate is limited by T_1 delay time generated in the Source Handshake function, which is defined according to the type of transceivers used. When the "High Speed" feature is enabled, T_1 = 2 microseconds is generated for the first byte transmitted after each true to false transition of ATN. For all subsequent bytes, T_1 = 500 ns. Refer to the Internal Counter section for an explanation of T_1 duration as a function of B_2 and of clock frequency.

B_3—Enable Active Low Interrupt: Setting this bit causes the polarity of the INT pin to be reversed, providing an output signal compatible with Intel's MCS-48® Family. Interrupt registers are not affected by this bit.

B_4—Enable RFD Holdoff on GET or DEC: Setting this bit causes RFD to be held false until the "VSCMD" auxiliary command is written after GET, SDC, and DCL commands. This allows the device to hold off the bus until it has completed a clear or trigger similar to an unrecognized command.

PARALLEL POLL PROTOCOL

Writing a 011USP$_3$P$_2$P$_1$ into the Auxiliary Mode Register will enable (U = 0) or disable (U = 1) the 8291A for a parallel poll. When U = 0, this command is the "lpe" (local poll enable) local message as defined in IEEE-488. The "S" bit is the sense in which the 8291A is enabled; only if the Parallel Poll Flag ("ist" local message) matches this bit will the Parallel Poll Response, PPR_N, be sent true (Response = S + ist). The bits $P_3P_2P_1$ specify which of the eight data lines PPR_N will be sent over. Thus, once the 8291A has been configured for Parallel Poll, whenever it senses both EOI and ATN true, it will automatically compare its PP flag with the sense bit and send PPR_N true or false according to the comparison.

If a PP2* implementation is desired, the "lpe" and "ist" local messages are all that are needed. Typically, the user will configure the 8291A for Parallel Poll immediately after initialization. During normal operation the microprocessor will set or clear the Parallel Poll Flag (ist) according to the device's need for service. Consequently the 8291A will be set up to give the proper response to IDY (EOI • ATN) without directly involving the microprocessor.

If a PP1* implementation is desired, the undefined command features of the 8291A must be used. In PP1, the 8291A is indirectly configured for Parallel Poll by the active controller on the GPIB. The sequence at the 8291A being enabled or disabled remotely is as follows:

1. The PPC message is received and is loaded into the Command Pass Through Register as an undefined command. A CPT Interrupt is sent to the microprocessor; the handshake is automatically held off.
2. The microprocessor reads the CPT Register and sends VSCMD to the 8291A, releasing the handshake.
3. Having recieved an undefined primary command, the 8291A is set up to receive an undefined secondary command (the PPE or PPD message). This message is also received into the CPT Register, the handshake is held off, and the CPT interrupt is generated.
4. The microprocessor reads the PPE or PPD message and writes the command into the Auxiliary Mode Register (bit 7 should be cleared first). Finally, the microprocessor sends VSCMD and the handshake is released.

NOTE:
*As defined in IEEE Standard 488.

End of Sequence (EOS) Register

EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

EOS REGISTER

The EOS Register and its features offer an alternative to the "Send EOI" auxiliary command. A seven or eight bit byte (ASCII or binary) may be placed in the register to flag the end of a block or read. The type of EOS byte to be used is selected in Auxiliary Register bit A_4.

If the 8291A is a listener, and the "End on EOS Received" is enabled with bit A_2, then an END interrupt is generated in the Interrupt Status 1 Register whenever the byte in the Data-In Register matches the byte in the EOS Register.

If the 8291A is a talker, and the "Output EOI on EOS Sent" is enabled with bit A_3, then the EOI line is sent true with the next byte whenever the contents of the Data Out Register match the EOS register.

Reset Procedure

The 8291A is reset to an initialization state either by a pulse applied to its Reset pin, or by a reset auxiliary command (02H written into the Auxiliary Command Register). The following conditions are caused by a reset pulse (or local reset command):

1. A "pon" local message as defined by IEEE-488 is held true until the initialization state is released.
2. The Interrupt Status Registers are cleared (not Interrupt Enable Registers).
3. Auxiliary Registers A and B are cleared.
4. The Serial Poll Mode Register is cleared.
5. The Parallel Poll Flag is cleared.
6. The EOI bit in the Address Status Register is cleared.
7. N_F in the Internal Counter is set to 8 MHz. This setting causes the longest possible T_1 delay to be generated in the Source Handshake (16 μs for 1 MHz clock).
8. The rdy local message is sent.

The initialization state is released by an "immediate execute pon" command (00H written into the Auxiliary Command Register).

The suggested initialization sequence is:

1. Apply a reset pulse or send the reset auxiliary command.
2. Set the desired initial conditions by writing into the Interrupt Enable, Serial Poll Mode, Address Mode, Address 0/1, and EOS Registers. Auxiliary Registers A and B, and the internal counter should also be initialized.
3. Send the "immediate execute pon" auxiliary command to release the initialization state.
4. If a PP2 Parallel Poll implementation is to be used the "lpe" local message may be sent, enabling the 8291A for a Parallel Poll Response on an assigned line. (Refer to the section on Parallel Poll Protocol.)

Using DMA

The 8291A may be connected to the Intel® 8237 or 8257 DMA Controllers or the 8089 I/O Processor for DMA operation. The 8237 will be used to refer to any DMA controller. The DREQ pin of the 8291A requests a DMA byte transfer from the 8237. It is set by BO or BI flip flops, enabled by the DMAO and DMAI bits in the Interrupt Enable 2 Register. (After reading , the INT1 register BO and BI interrupts will be cleared but not BO and BI in DREQ equation.)

The $\overline{DACK}$ pin is driven by the 8237 in response to the DMA request. When $\overline{DACK}$ is true (active low) it sets $\overline{CS}$ = RS0 = RS1 = RS2 = 0 such that the $\overline{RD}$ and $\overline{WR}$ signals sent by the 8237 refer to the Data In and Data Out Registers. Also, the DMA request line is reset by $\overline{DACK}$ ($\overline{RD}$ + $\overline{WR}$).

DMA input sequence:

1. A data byte is accepted from the GPIB by the 8291A.
2. A BI interrupt is generated and DREQ is set.
3. $\overline{DACK}$ and $\overline{RD}$ are driven by the 8237, the contents of the Data In Register are transferred to the system bus, and DREQ is reset.
4. The 8291A sends RFD true on the GPIB and proceeds with the Acceptor Handshake protocol.

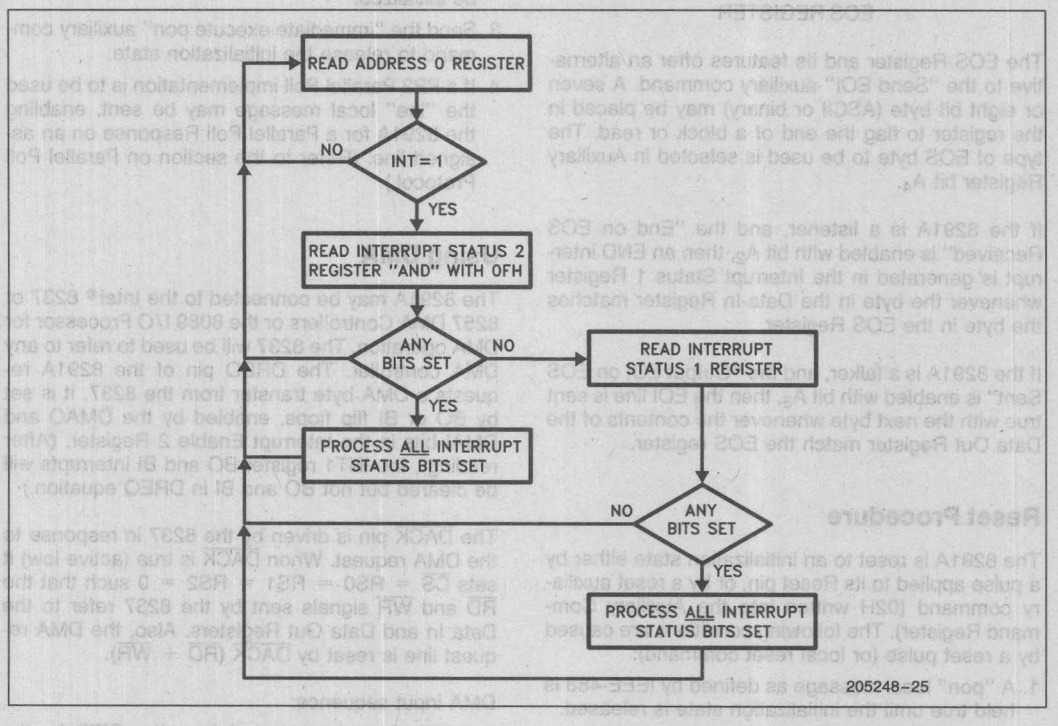
DMA output sequence:

1. A BO interrupt is generated (indicating that a byte should be output) and DREQ is asserted.
2. $\overline{DACK}$ and $\overline{WR}$ are driven by the 8237, a byte is transferred from the MCS bus into the Data Out Register, and DREQ is reset.
3. The 8291A sends DAV true on the GPIB and proceeds with the Source Handshake protocol.

It should be noted that each time the device is addressed (MTA + MLA + ton + lon), the Address Status Register should be read, and the 8237 should be initialized accordingly. (Refer to the 8237 or 8257 Data Sheets.)

Polling the 8291A

If polling is used to determine the 8291A's service needs, the CPU must poll the INT bit in the address 0 register. All relevant interrupt status bits must be enabled during initialization for them to affect the INT status bit. The following flow chart illustrates the recommended polling algorithm.



APPLICATION BRIEF

System Configuration

MICROPROCESSOR BUS CONNECTION

The 8291A is 8048/49, 8051, 8080/85, and 8086/88 compatible. The three address pins (RS_0, RS_1, and RS_2) should be connected to the non-multiplexed address bus (for example: A_8, A_9, A_{10}). In case of 8080, any address lines may be used. If the three lowest address bits are used (A_0, A_1, A_2), then they must be demultiplexed first.

EXTERNAL TRANSCEIVERS CONNECTION

The 8293 GPIB Transceiver interfaces the 8291A directly to the IEEE-488 bus. The 8291A and two 8293's can be configured as a talker/listener (see Figure 6) or with the 8292 as a talker/listener/controller (see Figure 7). Absolutely no active or passive external components are required to comply with the complete IEEE-488 electrical specification.

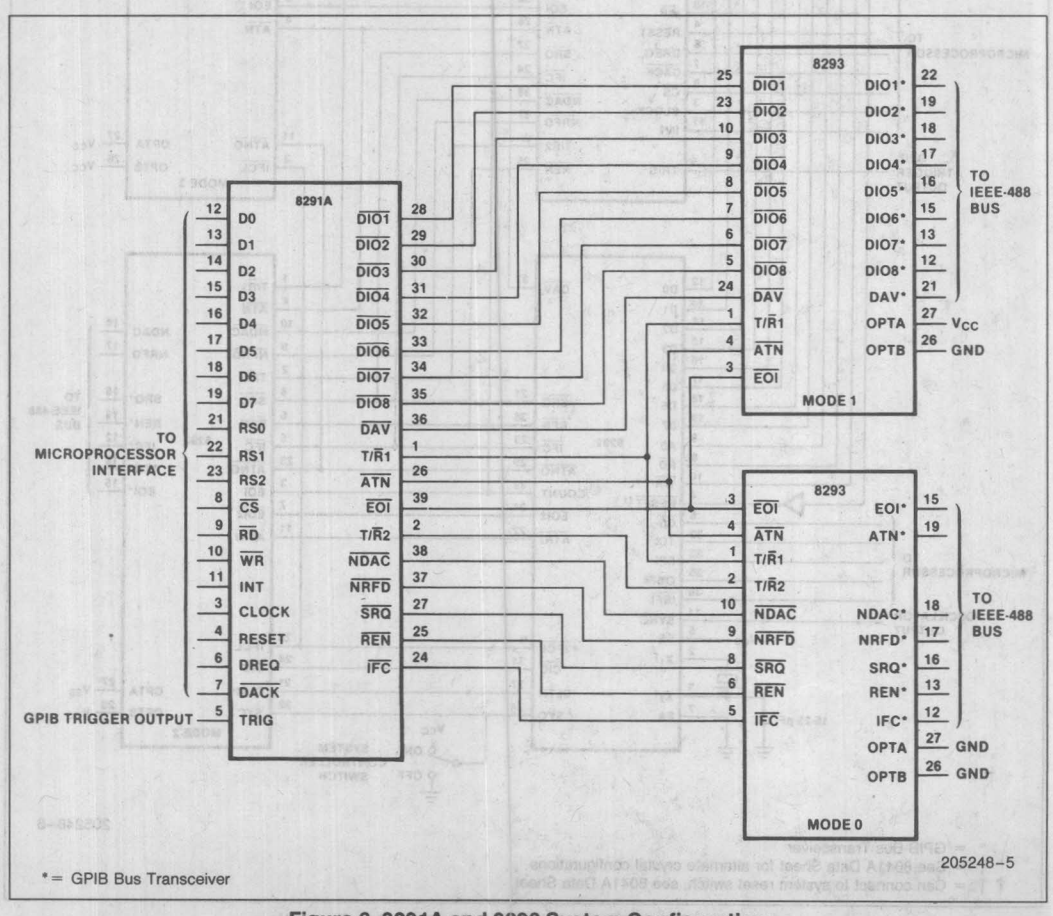
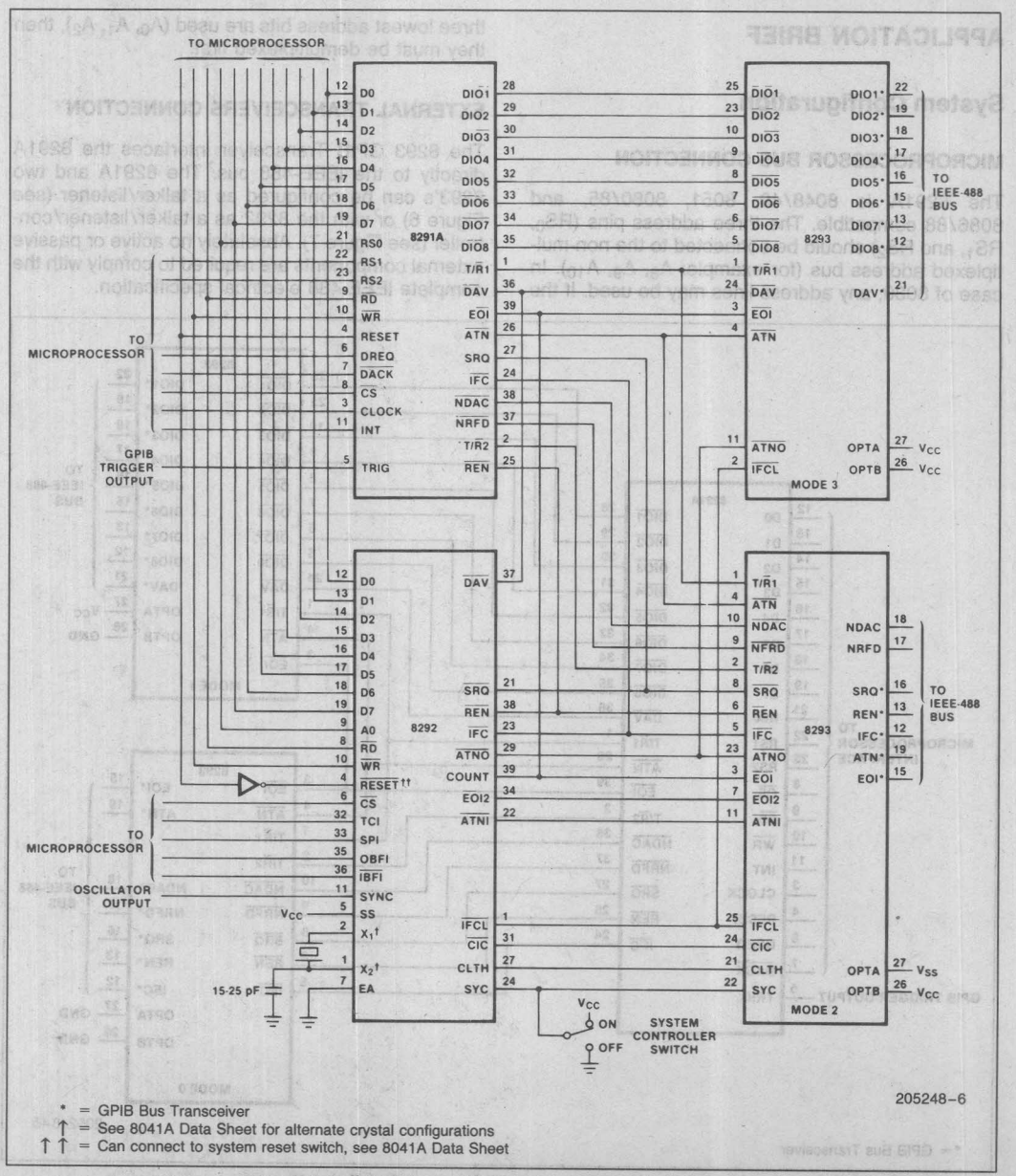


Figure 6. 8291A and 8293 System Configuration



* = GPIB Bus Transceiver
† = See 8041A Data Sheet for alternate crystal configurations
†† = Can connect to system reset switch, see 8041A Data Sheet

Figure 7. 8291A, 8292, and 8293 System Configuration

Start-Up Procedures

The following section describes the steps needed to initialize a typical 8291A system implementing a talker/listener interface and an 8291A/8292 system implementing a talker/listener/controller interface.

TALKER/LISTENER SYSTEM

Assume a general system configuration with the following features: (i) Polled system interface; (ii) Mode 1 addressing; (iii) same address for talker and listener; (iv) ASCII carriage return as the end-of-sequence (EOS) character; (v) EOI sent true with the last byte; and, (vi) 8 MHz clock.

Initialization. Initialization is accomplished with the following steps:

1. Pulse the RESET input or write 02H to the Auxiliary Mode Register.
2. Write 00H to the Interrupt Enable Registers 1 and 2. This disables interrupt and DMA.
3. Write 01H to the Address Mode Register to select Mode 1 addressing.
4. Write 28H to the Auxiliary Mode Register. This loads 8H to the Auxiliary Register A matching the 8 MHz clock input to the internal T1 delay counter to generate the delay meeting the IEEE spec.
5. Write the talker/listener address to the Address 0/1 register. The three most significant bits are zero.
6. Write an ASCII carriage return (0DH) to the EOS register.
7. Write 84H to the Auxiliary Mode Register to allow $\overline{EOI}$ to be sent true when the EOS character is sent.
8. Write 00H to the Auxiliary Mode Register. This writes the "Immediate Execute pon" message and takes the 8291A from the initialization state into the idle state. The 8291A will remain idle until the controller initiates some activity by driving $\overline{ATN}$ true.

Communication. The local CPU now polls the 8291A to determine which controller command has been received.

The controller addresses the 8291A by driving $\overline{ATN}$, placing MLA (My Listen Address) on the bus and driving $\overline{DAV}$. If the lower five bits of the MLA message match the address programmed into the Address 0/1 register, the 8291A is addressed to listen. It would be addressed to talk if the controller sent the MTA message instead of MLA.

The ADSC bit in the Interrupt Status 2 Register indicates that the 8291A has been addressed or unaddressed. The TA and LA bits in the Address Status Register indicate whether the 8291A is talker (TA = 1), listener (LA = 1), both (TA = LA = 1) or unaddressed (TA = LA = 0).

If the 8291A is addressed to listen, the local CPU can read the Data-In Register whenever the BI (Byte In) interrupt occurs in the Interrupt Status 1 Register. If the END bit in the same register is also set, either EOI or a data byte matching the pattern in the EOS register has been received.

In the talker mode, the CPU writes data into the Byte-Out Register on BO (Byte Out) true.

TALKER/LISTENER/CONTROLLER SYSTEM

Combined with the Intel 8292, the 8291A executes a complete IEEE-488-1978 controller function. The 8291A talks and listens via the data and handshake lines ($\overline{NRFD}$, $\overline{NDAC}$ and $\overline{DAV}$). The 8292 controls four of the five bus management lines ($\overline{IFC}$, $\overline{SRQ}$, $\overline{ATN}$ and $\overline{REN}$). $\overline{EOI}$, the fifth line, is shared. The 8291A drives and receives $\overline{EOI}$ when $\overline{EOI}$ is used as an end-of-block indicator. The 8292 drives $\overline{EOI}$ along with $\overline{ATN}$ during a parallel poll command.

Once again, assume a general system configuration with the following features: (i) Polled system interface; (ii) 8292 as the system controller and controller-in-charge; (iii) ASCII carriage return (0DH) as the EOS identifier; (iv) $\overline{EOI}$ sent with the last character; and, (v) an external buffer (8282) used to monitor the TCI line.

Initialization. In order to send a command across the GPIB, the 8292 has to drive $\overline{ATN}$, and the 8291A has to drive the data lines. Both devices therefore need initialization.

To initialize the 8292:

1. Pulse the RESET input. The 8292 will initially drive all outputs high. TCI, SPI, OBFI, IBFI and CLTH will then go low. The Interrupt Status, Interrupt Mask, Error Flag, Error Mask and Timeout registers will be cleared. The interrupt counter will be disabled and loaded with 255. The 8292 will then monitor the status of the SYC pin. If high, the 8292 will pulse IFC true for at least 100 μs in compliance with the IEEE-488-1978 standard. It will then take control by asserting $\overline{ATN}$.

To initialize the 8291A, the following is necessary:

1. Write 00H to Interrupt Enable registers 1 and 2. This disables interrupt and DMA.

2. With the 8292 as the controller-in-charge, it is impossible to address the 8292 via the GPIB. Therefore, the ton or lon modes of the 8291A must be used. To send commands, set the 8291A in the ton mode by writing 80H to the Address Mode Register.
3. Write 26H to the Auxiliary Mode Register to match the T1 data settling time to the 6 MHz clock input.
4. Write an ASCII carriage return (0DH) to the EOS Register.
5. Write 84H to the Auxiliary Mode Register in order to enable "Output EOI on EOS sent" and thus send EOI with the last character.
6. Write 00H—Immediate Execute pon—to the Auxiliary Mode Register to put the 8291A in the idle state.

Communication. Since the 8291A is in the ton mode, a BO interrupt is generated as soon as the immediate Execute pon command is written. The CPU writes the command into the Data Out Register, and repeats it on BO becoming true for as many commands as necessary. $\overline{ATN}$ remains continuously true unless the GTSB (Go To Standby) command is sent to the 8292.

$\overline{ATN}$ has to be false in order to send data rather than commands from the controller. To do this, the following steps are needed:

1. Enable the TCI interrupt if not already enabled.
2. Wait for IBF (Input Buffer Full) in the 8292 Interrupt Status Register to be reset.
3. Write the GTSB (F6H) command to the 8292 Command Field Register.
4. Read the 8282 and wait for TCI to be true.
5. Write the ton (80H) and pon (00H) command to the 8291A Address Mode Register and Auxiliary Mode Registers respectively.
6. Wait for the BO interrupt to be set in the 8291A.
7. Write the data to the 8291A Data-Out Register.

Identically, the user could command the controller to listen rather than talk. To do that, write lon (40H) instead of ton into the Address Mode Register. Then wait for BI rather than BO to go true. Read the data Register.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on Any Pin
With Respect to Ground .....−0.5V to +7V
Power Dissipation.....0.65 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS V_{CC} = 5V ±10%, T_A = 0°C to 70°C (Commercial)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	0.8	V	
V_{IH}	Input High Voltage	2	V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2 mA (4 mA for TR1 pin)
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA (−150 μA for SRQ pin)
V_{OH-INT}	Interrupt Output High Voltage	2.4 3.5		V V	I_{OH} = −400 μA I_{OH} = −50 μA
I_{IL}	Input Leakage		10	μA	V_{IN} = 0V to V_{CC}
I_{OFL}	Output Leakage Current		±10	μA	V_{OUT} = 0.45V, V_{CC}
I_{CC}	V_{CC} Supply Current		120	mA	T_A = 0°C

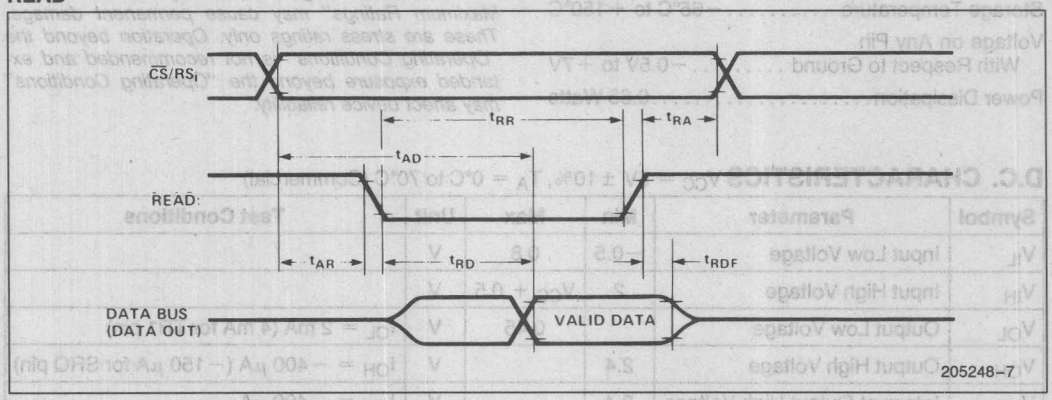
A.C. CHARACTERISTICS V_{CC} = 5V ±10%, T_A = 0°C to 70°C (Commercial)

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AR}	Address Stable Before $\overline{READ}$	0		ns	
t_{RA}	Address Hold After $\overline{READ}$	0		ns	
t_{RR}	$\overline{READ}$ Width	140		ns	
t_{AD}	Address Stable to Data Valid		250	ns	
t_{RD}	$\overline{READ}$ to Data Valid		100	ns	
t_{RDF}	Data Float After $\overline{READ}$	0	60	ns	
t_{AW}	Address Stable Before $\overline{WRITE}$	0		ns	
t_{WA}	Address Hold After $\overline{WRITE}$	0			
t_{WW}	$\overline{WRITE}$ Width	170		ns	
t_{DW}	Data Set Up Time to the Trailing Edge of WRITE	130		ns	
t_{WD}	Data Hold Time After $\overline{WRITE}$	0		ns	
t_{DKDR4}	$\overline{RD}$ ↓ or $\overline{WR}$ ↓ to DREQ ↓		130	ns	
t_{DKDA6}	$\overline{RD}$ ↓ to Valid Data (D_0–D_7)		200	ns	$\overline{DACK}$ ↓ to $\overline{RD}$ ↓ 0 ≤ t ≤ 50 ns

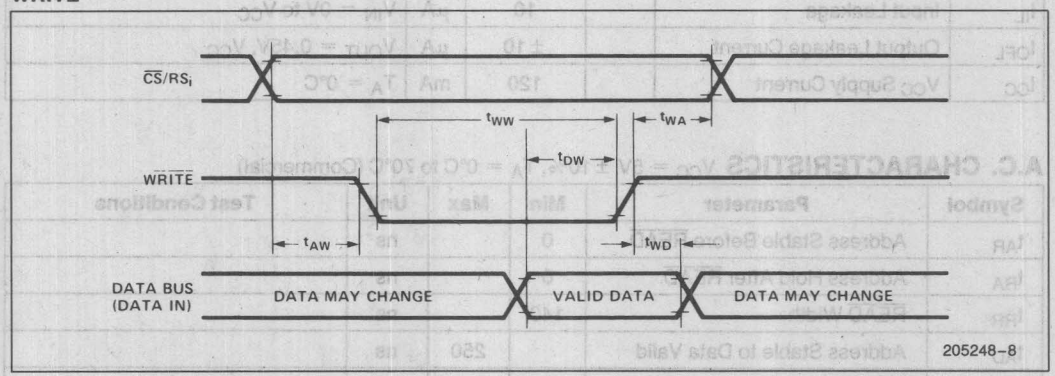
3

WAVEFORMS

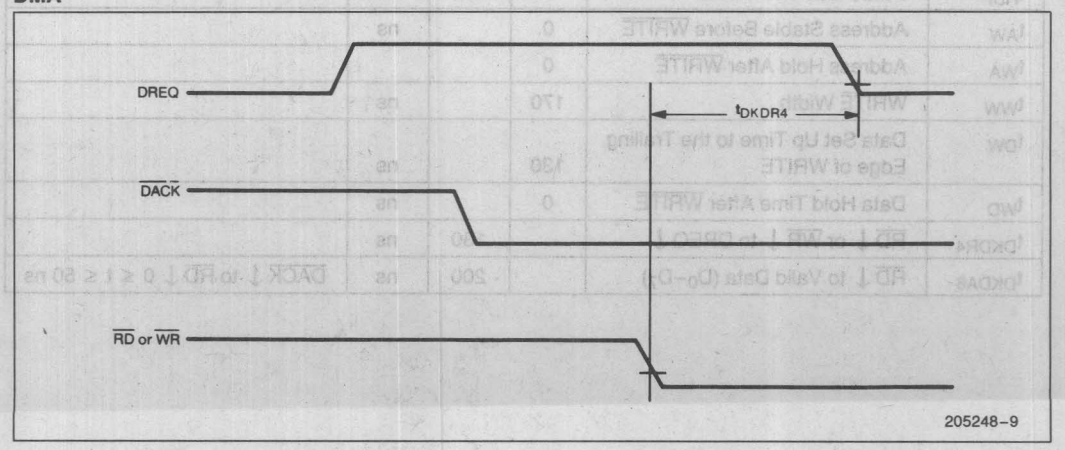
READ



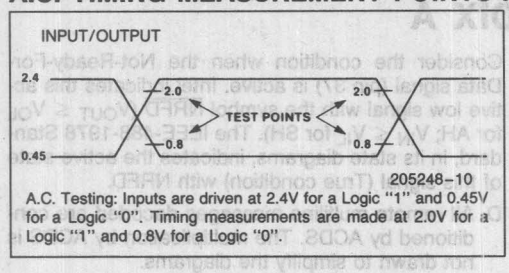
WRITE



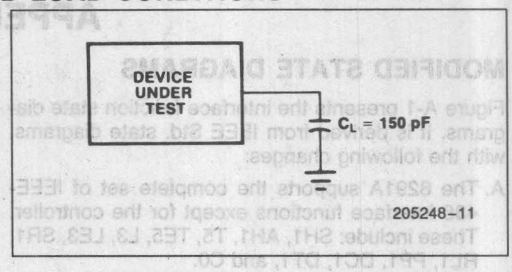
DMA



A.C. TIMING MEASUREMENT POINTS AND LOAD CONDITIONS



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".



GPIB TIMINGS(1)

Symbol	Parameter	Max	Units	Test Conditions
TEOT13(2)	$\overline{EOI}$ ↓ to TR1 ↑	135	ns	PPSS, ATN = 0.45V
TEOD16	$\overline{EOI}$ ↓ to $\overline{DIO}$ Valid	155	ns	PPSS, ATN = 0.45V
TEOT12	$\overline{EOI}$ ↑ to TR1 ↓	155	ns	PPSS, ATN = 0.45V
TATND4	$\overline{ATN}$ ↓ to $\overline{NDAC}$ ↓	155	ns	TACS, AIDS
TATT14	$\overline{ATN}$ ↓ to TR1 ↓	155	ns	TACS, AIDS
TATT24	$\overline{ATN}$ ↓ to TR2 ↓	155	ns	TACS, AIDS
TDVND3-C	$\overline{DAV}$ ↓ to $\overline{NDAC}$ ↑	650	ns	AH, CACS
TNDDV1	$\overline{NDAC}$ ↑ to $\overline{DAV}$ ↑	350	ns	SH, STRS
TNRDR1	$\overline{NRFD}$ ↑ to $\overline{DREQ}$ ↑	400	ns	SH
TDVDR3	$\overline{DAV}$ ↓ to $\overline{DREQ}$ ↑	600	ns	AH, LACS, ATN = 2.4V
TDVND2-C	$\overline{DAV}$ ↑ to $\overline{NDAC}$ ↓	350	ns	AH, LACS
TDVNR1-C	$\overline{DAV}$ ↑ to $\overline{NRFD}$ ↑	350	ns	AH, LACS, rdy = True
TRDNR3	$\overline{RD}$ ↓ to $\overline{NRFD}$ ↑	500	ns	AH, LACS
TWRD15	$\overline{WR}$ ↑ to $\overline{DIO}$ Valid	280	ns	SH, TACS, RS = 0.4V
TWREO5	$\overline{WR}$ ↑ to $\overline{EOI}$ Valid	350	ns	SH, TACS
TWRDV2	$\overline{WR}$ ↑ to $\overline{DAV}$ ↓	830 + t_{SYNC}	ns	High Speed Transfers Enabled, $N_F = f_C$, $t_{SYNC} = \frac{1}{2} \cdot f_C$

NOTES:
1. All GPIB timings are at the pins of the 8291A.
2. The last number in the symbol for any GPIB timing parameter is chosen according to the transition directions of the reference signals. The following table describes the numbering scheme.

↑ to ↑	1
↑ to ↓	2
↓ to ↑	3
↓ to ↓	4
↑ to VALID	5
↓ to VALID	6

3

APPENDIX A

MODIFIED STATE DIAGRAMS

Figure A-1 presents the interface function state diagrams. It is derived from IEEE Std. state diagrams, with the following changes:

A. The 8291A supports the complete set of IEEE-488 interface functions except for the controller. These include: SH1, AH1, T5, TE5, L3, LE3, SR1, RL1, PP1, DC1, DT1, and C0.

B. Addressing modes included in T, L state diagrams.

Note that in Mode 3, MSA, OSA are generated only after secondary address validity check by the microprocessor (APT interrupt).

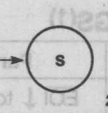
C. In these modified state diagrams, the IEEE-488-1978 convention of negative (low true) logic is followed. This should not be confused with the Intel pin- and signal-naming convention based on positive logic. Thus, while the state diagrams below carry low true logic, the signals described elsewhere in this data sheet are consistent with Intel notation and are based on positive logic.

		Convention	
Level	Logic	IEEE-488	Intel
0	T	DAV	$\overline{DAV}$
1	F	$\overline{DAV}$	DAV
0	T	NDAC	$\overline{NDAC}$
1	F	$\overline{NDAC}$	NDAC
0	T	NRFD	$\overline{NRFD}$
1	F	$\overline{NRFD}$	NRFD

Consider the condition when the Not-Ready-For-Data signal (pin 37) is active. Intel indicates this active low signal with the symbol $\overline{NRFD}$ ($V_{OUT} \leq V_{OL}$ for AH; $V_{IN} \leq V_{IL}$ for SH). The IEEE-488-1978 Standard, in its state diagrams, indicates the active state of this signal (True condition) with NRFD.

D. All remote multiline messages decoded are conditioned by ACDS. The multiplication by ACDS is not drawn to simplify the diagrams.

E. The symbol



205248-12

indicates:

1. When event X occurs, the function returns to state S.
2. X overrides any other transition condition in the function.

Statement 2 simplifies the diagram, avoiding the explicit use of X to condition all transitions from S to other states.

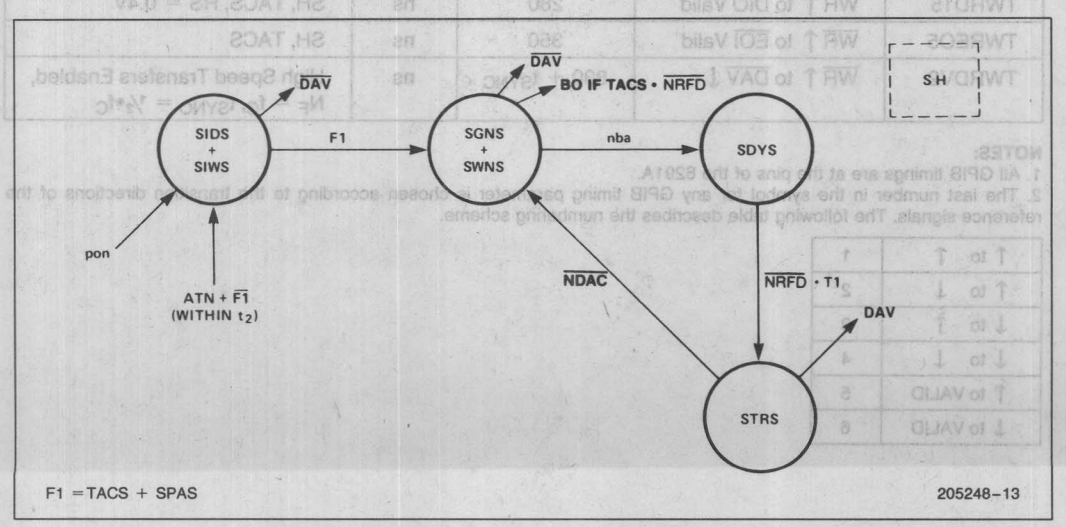
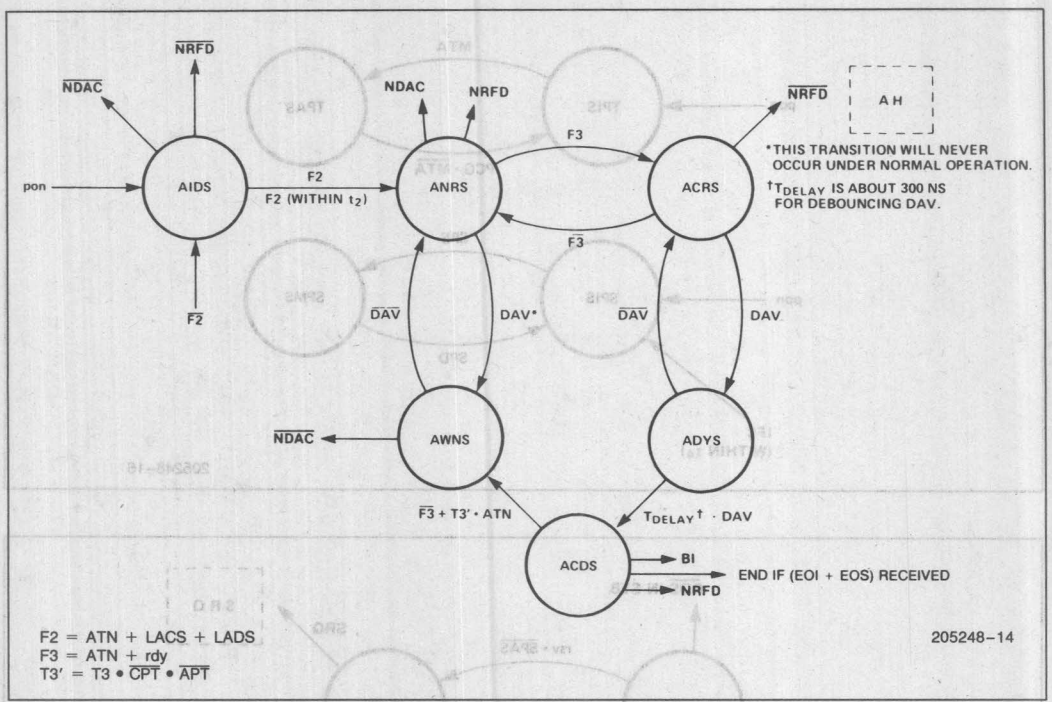


Figure A-1. 8291A State Diagrams



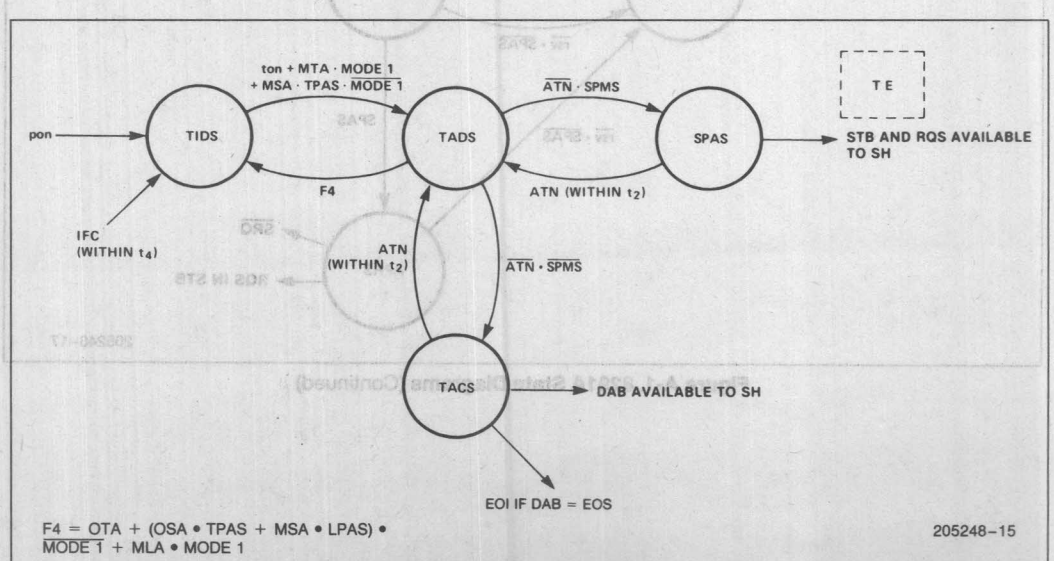
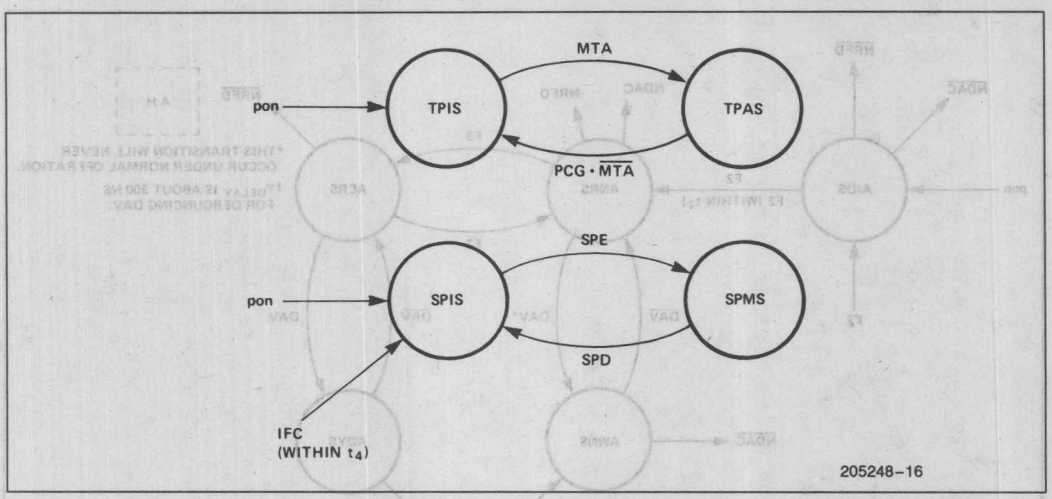


Figure A-1. 8291A State Diagrams (Continued)



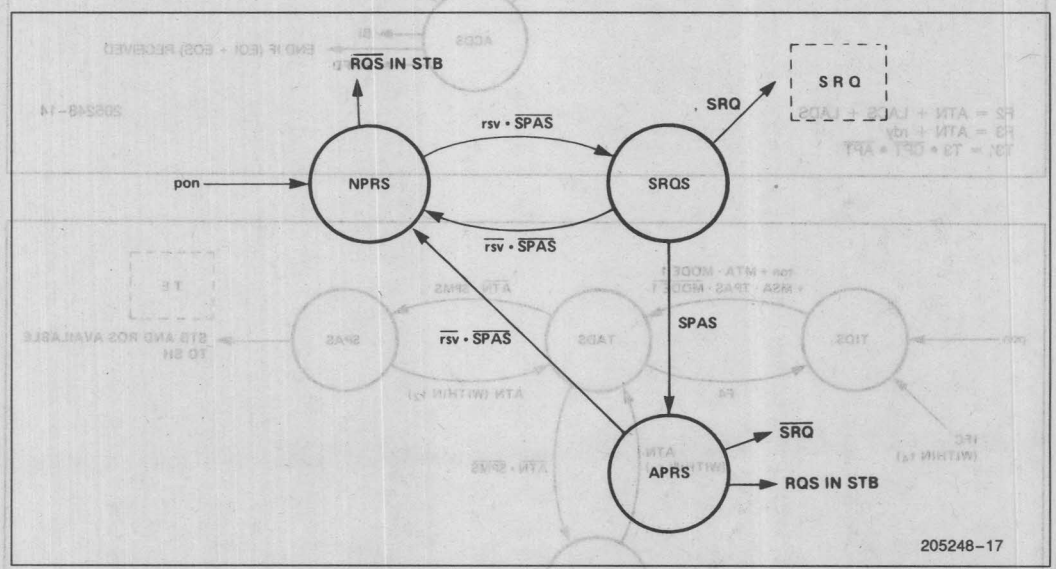
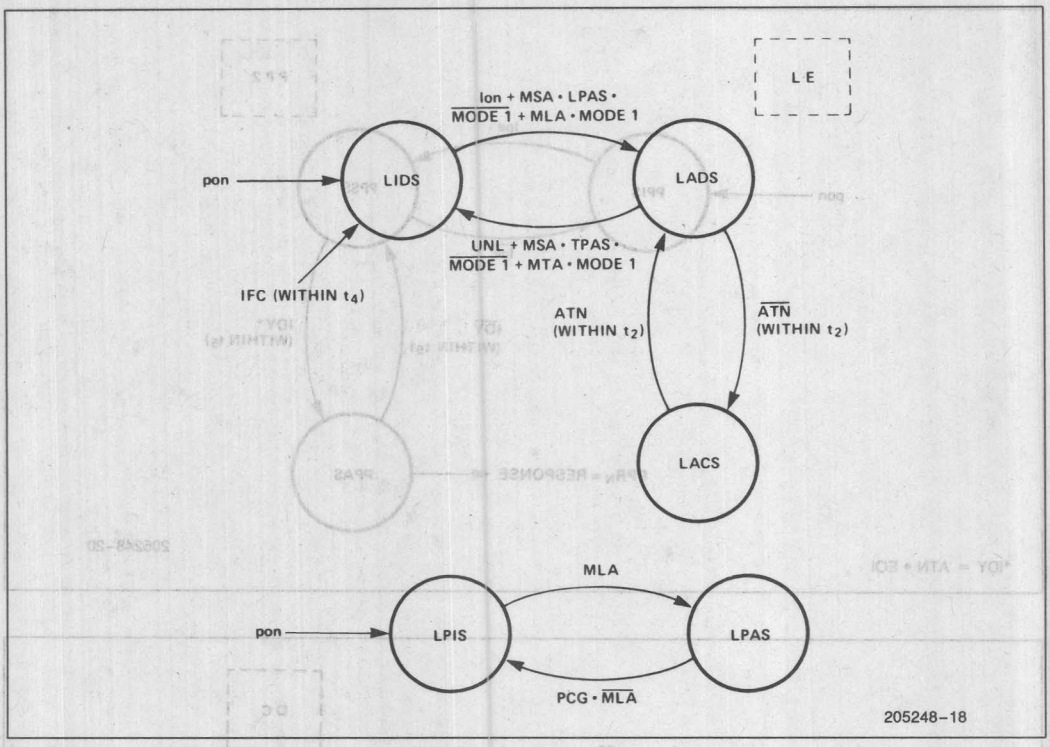


Figure A-1. 8291A State Diagrams (Continued)



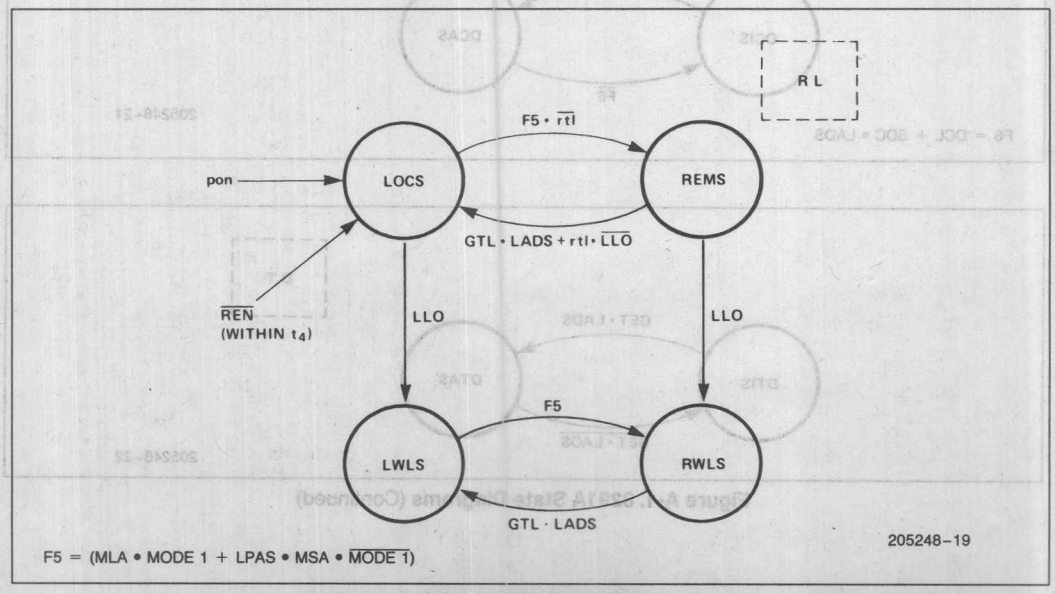


Figure A-1. 8291A State Diagrams (Continued)

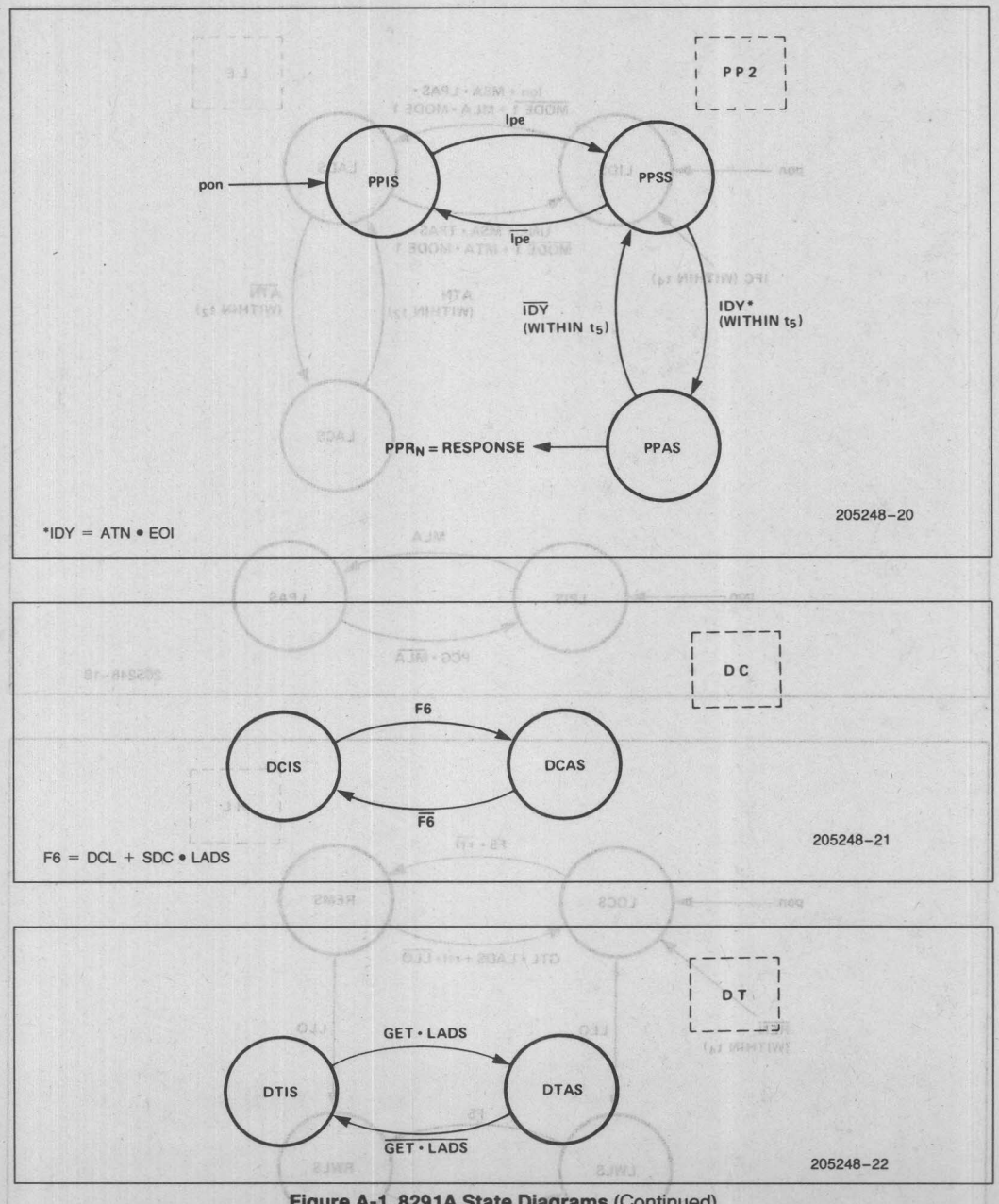


Figure A-1. 8291A State Diagrams (Continued)

APPENDIX B

Table B-1. IEEE 488 Time Values

Time Value Identifier(1)	Function (Applies to)	Description	Value
T_1	SH	Settling Time for Multiline Messages	≥ 2 μs(2)
t_2	LC, $\overline{IC}$, SH, AH, T, L	Response to ATN	≤ 200 ns
T_3	AH	Interface Message Accept Time(3)	> 0(4)
t_4	T, TE, L, LE, C, CE	Response to IFC or REN False	< 100 μs
t_5	PP	Response to ATN + EOI	≤ 200 ns
T_6	C	Parallel Poll Execution Time	≥ 2 μs
T_7	C	Controller Delay to Allow Current Talker to see ATN Message	≥ 500 ns
T_8	C	Length of IFC or REN False	> 100 μs
T_9	C	Delay for EOI(5)	≥ 1.5 μs(6)

NOTES:
1. Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.
2. If three-state drivers are used on the $\overline{DIO}$, $\overline{DAV}$, and $\overline{EOI}$ lines, T_1 may be:
   1. ≥ 1100 ns.
   2. Or ≥ 700 ns if it is known that within the controller ATN is driven by a three-state driver.
   3. Or ≥ 500 ns for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2).
   4. Or ≥ 350 ns for all subsequent bytes following the first sent after each false transition of ATN under conditions specified in Section 5.2.3 and warning note. See IEEE Standard 488.
3. Time required for interface functions to accept, not necessarily respond to interface messages.
4. Implementation dependent.
5. Delay required for $\overline{EOI}$, $\overline{NDAC}$, and $\overline{NRFD}$ signal lines to indicate valid states.
6. ≥ 600 ns for three-state drivers.

3

APPENDIX C
THE THREE-WIRE HANDSHAKE

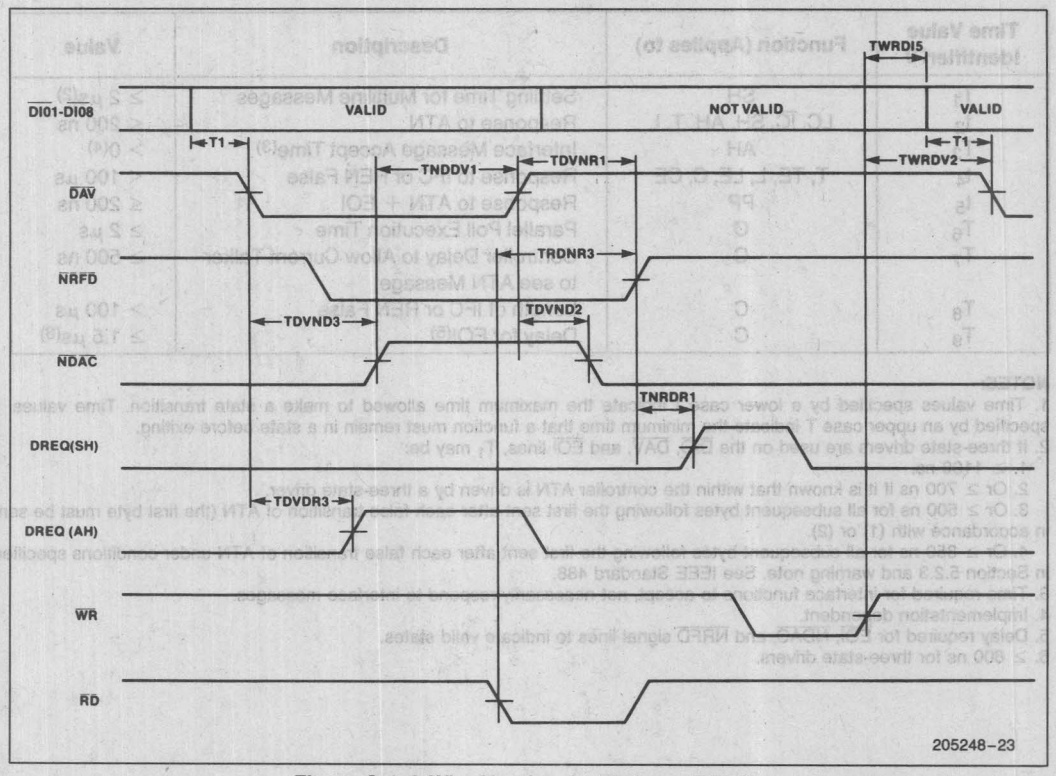
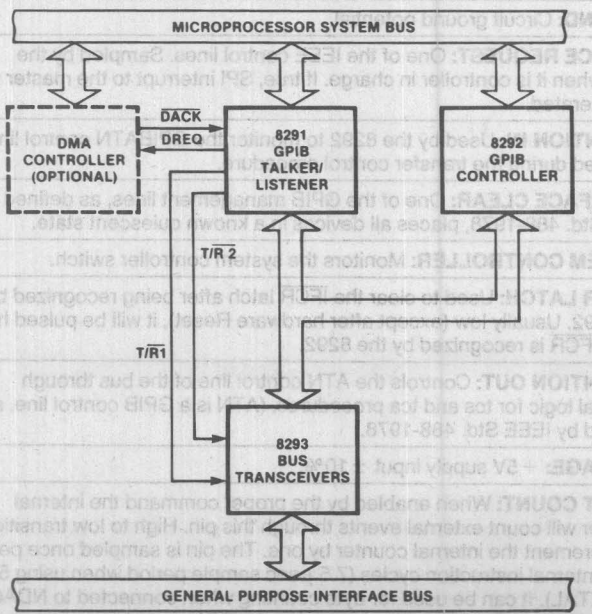


Figure C-1. 3-Wire Handshake Timing at 8291A

8292
GPIB CONTROLLER

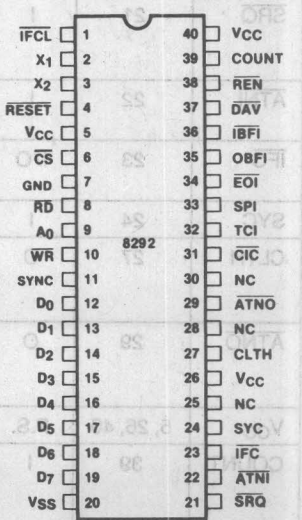
- **Complete IEEE Standard 488 Controller Function**
- **Interface Clear (IFC) Sending Capability Allows Seizure of Bus Control and/or Initialization of the Bus**
- **Responds to Service Requests (SRQ)**
- **Sends Remote Enable (REN), Allowing Instruments to Switch to Remote Control**
- **Complete Implementation of Transfer Control Protocol**
- **Synchronous Control Seizure Prevents the Destruction of Any Data Transmission in Progress**
- **Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller**

The 8292 GPIB Controller is a microprocessor-controlled chip designed to function with the 8291 GPIB Talker/Listener to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a preprogrammed Intel® 8041A.



205250-1

Figure 1. 8291, 8292 Block Diagram



205250-2

Figure 2. Pin Configuration

November 1986
Order Number: 205250-002

Table 1. Pin Description

Symbol	Pin Number	Type	Name and Function
$\overline{IFCL}$	1	I	**IFC RECEIVED (LATCHED):** The 8292 monitors the IFC Line (when not system controller) through this pin.
X_1, X_2	2, 3	I	**CRYSTAL INPUTS:** Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
$\overline{RESET}$	4	I	**RESET:** Used to initialize the chip to a known state during power on.
$\overline{CS}$	6	I	**CHIP SELECT INPUT:** Used to select the 8292 from other devices on the common data bus.
$\overline{RD}$	8	I	**READ ENABLE:** Allows the master CPU to read from the 8292.
A_0	9	I	**ADDRESS LINE:** Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.
$\overline{WR}$	10	I	**WRITE ENABLE:** Allows the master CPU to write to the 8292.
SYNC	11	O	**SYNC:** 8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL ÷ 15.
D_0–D_7	12–19	I/O	**DATA:** 8 bidirectional lines used for communication between the central processor and the 8292's data bus buffers and status register.
V_{SS}	7, 20	P.S.	**GROUND:** Circuit ground potential.
$\overline{SRQ}$	21	I	**SERVICE REQUEST:** One of the IEEE control lines. Sampled by the 8292 when it is controller in charge. If true, SPI interrupt to the master will be generated.
$\overline{ATNI}$	22	I	**ATTENTION IN:** Used by the 8292 to monitor the GPIBATN control line. If is used during the transfer control procedure.
$\overline{IFC}$	23	I/O	**INTERFACE CLEAR:** One of the GPIB management lines, as defined by IEEE Std. 488-1978, places all devices in a known quiescent state.
SYC	24	I	**SYSTEM CONTROLLER:** Monitors the system controller switch.
CLTH	27	O	**CLEAR LATCH:** Used to clear the $\overline{IFCR}$ latch after being recognized by the 8292. Usually low (except after hardware Reset), it will be pulsed high when $\overline{IFCR}$ is recognized by the 8292.
$\overline{ATNO}$	29	O	**ATTENTION OUT:** Controls the ATN control line of the bus through external logic for tcs and tca procedures. (ATN is a GPIB control line, as defined by IEEE Std. 488-1978.)
V_{CC}	5, 26, 40	P.S.	**VOLTAGE:** +5V supply input ±10%.
COUNT	39	I	**EVENT COUNT:** When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5 μsec sample period when using 5 MHz XTAL). It can be used for byte counting when connected to NDAC, or for block counting when connected to the EOI.
$\overline{REN}$	38	O	**REMOTE ENABLE:** The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1978.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{DAV}$	37	I/O	**DATA VALID:** Used during parallel poll to force the 8291 to accept the parallel poll status bit. It is also used during the tcs procedure.
$\overline{IBFI}$	36	O	**INPUT BUFFER NOT FULL:** Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.
OBFI	36	O	**OUTPUT BUFFER FULL:** Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.
$\overline{EO12}$	34	I/O	**END OR IDENTIFY:** One of the GPIB management lines, as defined by IEEE Std. 488-1978. Used with ATN as Identify Message during parallel poll.
SPI	33	O	**SPECIAL INTERRUPT:** Used as an interrupt on events not initiated by the central processor.
TCI	32	O	**TASK COMPLETE INTERRUPT:** Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus buffer.
CIC	31	O	**CONTROLLER IN CHARGE:** Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the GPIB bus.

FUNCTIONAL DESCRIPTION

The 8292 is an Intel 8041A which has been programmed as a GPIB Controller Interface element. It is used with the 8291 GPIB Talker/Listener and two 8293 GPIB Transceivers to form a complete IEEE-488 Bus Interface for a microprocessor. The electrical interface is performed by the transceivers, data transfer is done by the talker/listener, and control of the bus is done by the 8292. Figure 3 is a typical controller interface using Intel's GPIB peripherals.

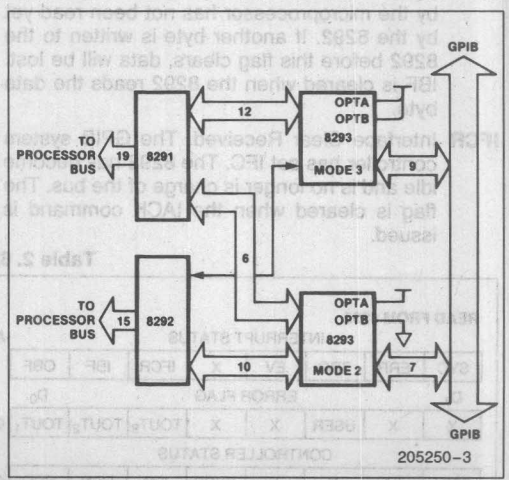


Figure 3. Talker/Listener/Controller Configuration

The internal RAM in the 8041A is used as a special purpose register bank for the 8292. Most of these registers (except for the interrupt flag) can be accessed through commands to the 8292. Table 2 identifies the registers used by the 8292 and how they are accessed.

Interrupt Status Register

SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF
D_7							D_0

The 8292 can be configured to interrupt the microprocessor on one of several conditions. Upon receipt of the interrupt the microprocessor must read the 8292 interrupt status register to determine which event caused the interrupt, and then the appropriate subroutine can be performed. The interrupt status register is read with A_0 high. With the exception of OBF and IBF, these interrupts are enabled or disabled by the SPI interrupt mask. OBF and IBF have their own bits in the interrupt mask (OBFI and $\overline{\text{IBFI}}$).

OBF Output Buffer Full. A byte is waiting to be read by the microprocessor. This flag is cleared when the output data bus buffer is read.

IBF Input Buffer Full. The byte previously written by the microprocessor has not been read yet by the 8292. If another byte is written to the 8292 before this flag clears, data will be lost. IBF is cleared when the 8292 reads the data byte.

IFCR Interface Clear Received. The GPIB system controller has set IFC. The 8292 has become idle and is no longer is charge of the bus. The flag is cleared when the IACK command is issued.

EV Event Counter Interrupt. The requested number of blocks of data byte has been transferred. The EV interrupt flag is cleared by the IACK command.

SRQ Service Request. Notified the 8292 that a service request (SRQ) message has been received. It is cleared by the IACK command.

ERR Error occurred. The type of error can be determined by reading the error status register. This interrupt flag is cleared by the IACK command.

SYC System Controller Switch Change. Notifies the processor that the state of the system controller switch has changed. The actual state is contained in the GPIB Status Register. This flag is cleared by the IACK command.

Interrupt Mask Register

1	SPI	TCI	SYC	OBFI	$\overline{\text{IBFI}}$	0	SRQ
D_7							D_0

The Interrupt Mask Register is used to enable features and to mask the SPI and TCI interrupts. The flags in the Interrupt Status Register will be active even when masked out. The Interrupt Mask Register is written when A_0 is low and reset by the RINM command. When the register is read, D_1 and D_7 are undefined. An interrupt is enabled by setting the corresponding register bit.

SRQ Enable interrupts on SRQ received.

$\overline{\text{IBFI}}$ Enable interrupts on input buffer empty.

OBFI Enable interrupts on output buffer full.

Table 2. 8292 Registers

READ FROM 8292

INTERRUPT STATUS								A_0
SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF	1
D_7	ERROR FLAG						D_0	
X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$	0*
CONTROLLER STATUS								
CSBS	CA	X	X	SYCS	IFC	REN	SRQ	0*
GPIB (BUS) STATUS								
REN	DAV	EOI	X	SYC	IFC	ATN	SRQ	0*
EVENT COUNTER STATUS								
D	D	D	D	D	D	D	D	0*
TIME OUT STATUS								
D	D	D	D	D	D	D	D	0*

WRITE TO 8292

INTERRUPT MASK								A_0
1	SPI	TCI	SYC	OBFI	$\overline{\text{IBFI}}$	0	SRQ	0
D_7	ERROR MASK						D_0	
0	0	USER	0	0	$TOUT_3$	$TOUT_2$	$TOUT_1$	0
COMMAND FIELD								
1	1	1	OP	C	C	C	C	1
EVENT COUNTER								
D	D	D	D	D	D	D	D	0*
TIME OUT								
D	D	D	D	D	D	D	D	0*

NOTE: These registers are accessed by a special utility command, see page 7.

SYC Enable interrupts on a change in the system controller switch.

TCI Enable interrupts on the task completed.

SPI Enable interrupts on special events.

NOTE:
The event counter is enabled by the GSEC command, the error interrupt is enabled by the error mask register, and IFC cannot be masked (it will always cause an interrupt).

Controller Status Register

CSBS	CA	X	X	SYCS	IFC	REN	SRQ
D_7							D_0

The Controller Status Register is used to determine the status of the controller function. This register is accessed by the RCST command.

SRQ Service Request line active (CSRS).

REN Sending Remote Enable.

IFC Sending or receiving interface clear.

SYCS System Controller Switch Status (SACS).

CA Controller Active (CACS + CAWS + CSWS).

CSBS Controller Stand-by State (CSBS, CA) = (0,0)—Controller Idle.

GPIB Bus Status Register

REN	DAV	EOI	X	SYC	IFC	ATNI	SRQ
D_7							D_0

This register contains GPIB bus status information. It can be used by the microprocessor to monitor and manage the bus. The GPIB Bus Register can be read using the RBST command.

Each of these status bits reflect the current status of the corresponding pin on the 8292.

SRQ Service Request

ATNI Attention In

IFC Interface Clear

SYC System Controller Switch

EOI End or Identify

DAV Data Valid

REN Remote Enable

Event Counter Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

The Event Counter Register contains the initial value for the event counter. The counter can count pulses on pin 39 of the 8292 (COUNT). It can be connected to EOI or NDAC to count blocks or bytes respectively during standby state. A count of zero equals 256. This register cannot be read, and is written using the WEVC command.

Event Counter Status Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

This register contains the current value in the event counter. The event counter counts back from the initial value stored in the Event Counter Register to zero and then generates an Event Counter Interrupt. This register cannot be written and can be read using a REVC command.

Time Out Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

The Time Out Register is used to store the time used for the time out error function. See the individual timeouts (TOUT1, 2, 3) to determine the units of this counter. This Time Out Register cannot be read, and it is written with the WTOUT command.

Time Out Status Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

This register contains the current value in the time out counter. The time out counter decrements from the original value stored in the Time Out Register. When zero is reached, the appropriate error interrupt is generated. If the register is read while none of the time out functions are active, the register will contain the last value reached the last time a function was active. The Time Out Status Register cannot be written, and it is read with RTOUT command.

Error Flag Register

X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$

Four errors are flagged by the 8292 with a bit in the Error Flag Register. Each of these errors can be masked by the Error Mask Register. The Error Flag Register cannot be written, and it is read by the IACK command when the error flag in the Interrupt Status Register is set.

TOUT1 Time Out Error 1 occurs when the current controller has not stopped sending ATN after receiving the TCT message for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t_{CY}. After flagging the error, the 8292 will remain in a loop trying to take control until the current controller stops send-

ing ATN or a new command is written by the microprocessor. If a new command is written, the 8292 will return to the loop after executing it.

TOUT2 Time Out Error 2 occurs when the transmission between the addressed talker and listener has not started for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 45 t_{CY}. This feature is only enabled when the controller is in the CSBS state.

TOUT3 Time Out Error 3 occurs when the handshake signals are stuck and the 8292 is not succeeding in taking control synchronously for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t_{CY}. The 8292 will continue checking $\overline{ATNI}$ until it becomes true or a new commnand is received. After performing the new command, the 8292 will return to the $\overline{ATNI}$ checking loop.

USER User error occurs when request to assert IFC or REN was received and the 8292 was not the system controller.

Error Mask Register

0	0	USER	0	0	$TOUT_3$	$TOUT_2$	$TOUT_1$
D_7							D_0

The Error Mask Register is used to mask the interrupt from a particular type of error. Each type of error interrupt is enabled by setting the corresponding bit in the Error Mask Register. This register can be read with the RERM command and written with A_0 low.

Command Register

1	1	1	OP	C	C	C	C
D_7							D_0

Commands are performed by the 8292 whenever a byte is written with A_0 high. There are two categories of commands distinguished by the OP bit (bit 4). The first category is the operation command (OP = 1). These commands initiate some action on the interface bus. The second category is the utility command (OP = 0). These commands are used to aid the communication between the processor and the 8292.

OPERATION COMMANDS

Operation commands initiate some action on the GPIB interface bus. It is using these commands that the control functions such as polling, taking and passing control, and system controller functions are performed.

F0—SPCNI—Stop Counter Interrupts

This command disables the internal counter interrupt so that the 8292 will stop interrupting the master on event counter underflows. However, the counter will continue counting and its contents can still be used.

F1—GIDL—Go To Idle

This command is used during the transfer of control procedure while transferring control to another controller. The 8292 will respond to this command only if it is in the active state. $\overline{ATNO}$ will go high, and $\overline{CIC}$ will be high so that this 8292 will no longer be driving the ATN line on the GPIB interface bus. TCI will be set upon completion.

F2—RST—Reset

This command has the same effect as asserting the external reset on the 8292. For details, refer to the reset procedure described later.

F3—RSTI—Reset Interrupts

This command resets any pending interrupts and clears the error flags. The 8292 will not return to any loop it was in (such as from the time out interrupts).

F4—GSEC—Go To Standby, Enable Counting

The function causes $\overline{ATNO}$ to go high and the counter will be enabled. If the 8292 was not the active controller, this command will exit immediately. If the 8292 is the active controller, the counter will be loaded with the value stored in the Event Counter Register, and the internal interrupt will be enabled so that when the counter reaches zero, the SPI interrupt will be generated. SPI will be generated every 256 counts thereafter until the controller exits the standby state or the SPCNI command is written. An initial count of 256 (zero in the Event Counter Register) will be used if the WEVC command is not executed. If the data transmission does not start, a TOUT2 error will be generated.

F5—EXPP—Execute Parallel Poll

This command initiates a parallel poll by asserting EOI when ATN is already active. TCI will be set at the end of the command. The 8291 should be previously configured as a listener. Upon detection of DAV true, the 8291 enters ACDS and latches the parallel poll response (PPR) byte into its data in register. The master will be interrupted by the 8291 BI interrupt when the PPR byte is available. No interrupts except the $\overline{IBFI}$ will be generated by the 8292. The 8292 will respond to this command only when it is the active controller.

F6—GTSB—Go To Standby

If the 8292 is the active controller, $\overline{ATNO}$ will go high then TCI will be generated. If the data transmission does not start, a TOUT2 error will be generated.

F7—SLOC—Set Local Mode

If the 8292 is the system controller, then REN will be asserted false and TCI will be set true. If it is not the system controller, the User Error bit will be set in the Error Flag Register.

F8—SREM—Set Interface To Remote Control

This command will set REN true and TCI true if this 8292 is the system controller. If not, the User Error bit will be set in the Error Flag Register.

F9—ABORT—Abort All Operation, Clear Interface

This command will cause IFC to be asserted true for at least 100 μsec if this 8292 is the system controller. If it is in CIDS, it will take control over the bus (see the TCNTR command).

FA—TCNTR—Take Control

The transfer of control procedure is coordinated by the master with the 8291 and 8292. When the master receives a TCT message from the 8291, it should issue the TCNTR command to the 8292. The following events occur to take control:

1) The 8292 checks to see if it is in CIDS, and if not, it exits.
2) Then $\overline{ATNI}$ is checked until it becomes high. If the current controller does not release ATN for the time specified by the Time Out Register, then a TOUT1 error is generated. The 8292 will return to this loop after an error or any command except the RST and RSTI commands.
3) After the current controller releases ATN, the 8292 will assert $\overline{ATNO}$ and $\overline{CIC}$ low.
4) Finally, the TCI interrupt is generated to inform the master that it is in control of the bus.

FC—TCASY—Take Control Asynchronously

TCAS transfers the 8292 from CSBS to CACS independent of the handshake lines. If a bus hangup is detected (by an error flag), this command will force the 8292 to take control (asserting ATN) even if the AH function is not in ANRS (Acceptor Not Ready State). This command should be used very carefully since it may cause the loss of a data byte. Normally, control should be taken synchronously. After checking the controller function for being in the CSBS (else it will exit immediately), $\overline{ATNO}$ will go low, and a TCI interrupt will be generated.

FD—TCSY—Take Control Synchronously

There are two different procedures used to transfer the 8292 from CSBS to CACS depending on the state of the 8291 in the system. If the 8291 is in "continuous AH cycling" mode (Aux. Reg. A0 = A1 = 1), then the following procedures should be followed:

1) The master microprocessor stops the continuous AH cycling mode in the 8291;
2) The master reads the 8291 Interrupt Status 1 Register;
3) If the END bit is set, the master sends the TCSY command to the 8292;
4) If the END bit was not set, the master reads the 8291 Data In Register and then waits for another BI interrupt from the 8291. When it occurs, the master sends the 8292 the TCSY command.

If the 8291 is not in AH cycling mode, then the master just waits for a BI interrupt and then sends the TCSY command. After the TCSY command has been issued, the 8292 checks for $\overline{CSBS}$. If $\overline{CSBS}$, then it exits the routine. Otherwise, it then checks the DAV bit in the GPIB status. When DAV becomes false, the 8292 will wait for at least 1.5 μsec. (T10) and then $\overline{ATNO}$ will go low. If DAV does not go low, a TOUT3 error will be generated. If the 8292 successfully takes control, it sets TCI true.

FE—STCNI—Start Counter Interrupts

This command enables the internal counter interrupt. The counter is enabled by the GSEC command.

UTILITY COMMANDS

All these commands are either Read or Write to registers in the 8292. Note that writing to the Error Mask Register and the Interrupt Mask Register are done directly.

E1—WTOUT—Write To Time Out Register

The byte written to the data bus buffer (with $A_0 = 0$) following this command will determine the time used for the time out function. Since this function is implemented in software, this will not be an accurate time measurement. This feature is enable or disable by the Error Mask Register. No interrupts except for the $\overline{IBFI}$ will be generated upon completion.

E2—WEVC—Write To Event Counter

The byte written to the data bus buffer (with $A_0 = 0$) following this command will be loaded into the Event Counter Register and the Event Counter Status for byte counting of EOI counting. Only $\overline{IBFI}$ will indicate completion of this command.

E3—REVC—Read Event Counter Status

This command transfers the contents of the Event Counter into the data bus buffer. A TCI is generated when the data is available in the data bus buffer.

E4—RERF—Read Error Flag Register

This command transfers the contents of the Error Flag Register into the data bus buffer. A TCI is generated when the data is available.

E5—RINM—Read Interrupt Mask Register

This command transfers the contents of the Interrupt Mask Register into the data bus buffer. This register is available to the processor so that it does not need to store this information elsewhere. A TCI is generated when the data is available in the data bus buffer.

E6—RCST—Read Controller Status Register

This command transfers the contents of the Controller Status Register into the data bus buffer and a TCI interrupt is generated.

E7—RBST—Read GPIB Bus Status Register

This command transfers the contents of the GPIB Bus Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

E9—RTOUT—Read Time Out Status Register

This command transfers the contents of the Time Out Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.

EA—RERM—Read Error Mask Register

This command transfers the contents of the Error Mask Register to the data bus buffer so that the processor does not need to store this information elsewhere. A TCI interrupt is generated when the data is available.

Interrupt Acknowledge

SYC	ERR	SRQ	EV	1	IFCR	1	1
D_7							D_0

Each named bit in an Interrupt Acknowledge (IACK) corresponds to a flag in the Interrupt Status Register. When the 8292 receives this command, it will clear the SPI and the corresponding bits in the Interrupt Status Register. If not all the bits were cleared, then the SPI will be set true again. If the error flag is not acknowledged by the IACK command, then the Error Flag Register will be transferred to the data bus buffer, and a TCI will be generated.

NOTE:
XXXX1X11 is an undefined operation or utility command, so no conflict exists between the IACK operation and utility commands.

SYSTEM OPERATION

8292 To Master Processor Interface

Communication between the 8292 and the Master Processor can be either interrupt based communication or based upon polling the interrupt status register in predetermined intervals.

Interrupt Based Communication

Four different interrupts are available from the 8292:

OBFI Output Buffer Full Interrupt

$\overline{IBFI}$ Input Buffer Not Full Interrupt

TCI Task Completed Interrupt

SPI Special Interrupt

Each of the interrupts is enabled or disabled by a bit in the interrupt mask register. Since OBFI and $\overline{IBFI}$ are directly connected to the OBF and IBF flags, the master can write a new command to the input data bus buffer as soon as the previous command has been read.

The TCI interrupt is useful when the master is sending commands to the 8292. The pending TCI will be cleared with each new command written to the 8292. Commands sent to the 8292 can be divided into two major groups:

1) Commands that require response back from the 8292 to the master, e.g., reading register.
2) Commands that initiate some action or enable features but do not require response back from the 8292, e.g., enable data bus buffer interrupts.

With the first group, the TCI interrupt will be used to indicate that the required response is ready in the data bus buffer and the master may continue and read it. With the second group, the interrupt will be used to indicate completion of the required task, so that the master may send new commands.

The SPI should be used when immediate information or special events is required (see the Interrupt Status Register).

"Polling Status" Based Communication

When interrupt based communication is not desired, all interrupts can be masked by the interrupt mask register. The communication with the 8292 is based upon sequential poll of the interrupt status register. By testing the OBF and IBF flags, the data bus buffer status is determined while special events are determined by testing the other bits.

Receiving IFC

The IFC pulse defined by the IEEE-488 standard is at least 100 μsec. In this time, all operation on the bus should be aborted. Most important, the current controller (the one that is in charge at that time) should stop sending ATN or EOI. Thus, IFC must externally gate $\overline{CIC}$ (controller in charge) and $\overline{ATNO}$ to ensure that this occurs.

Reset and Power Up Procedure

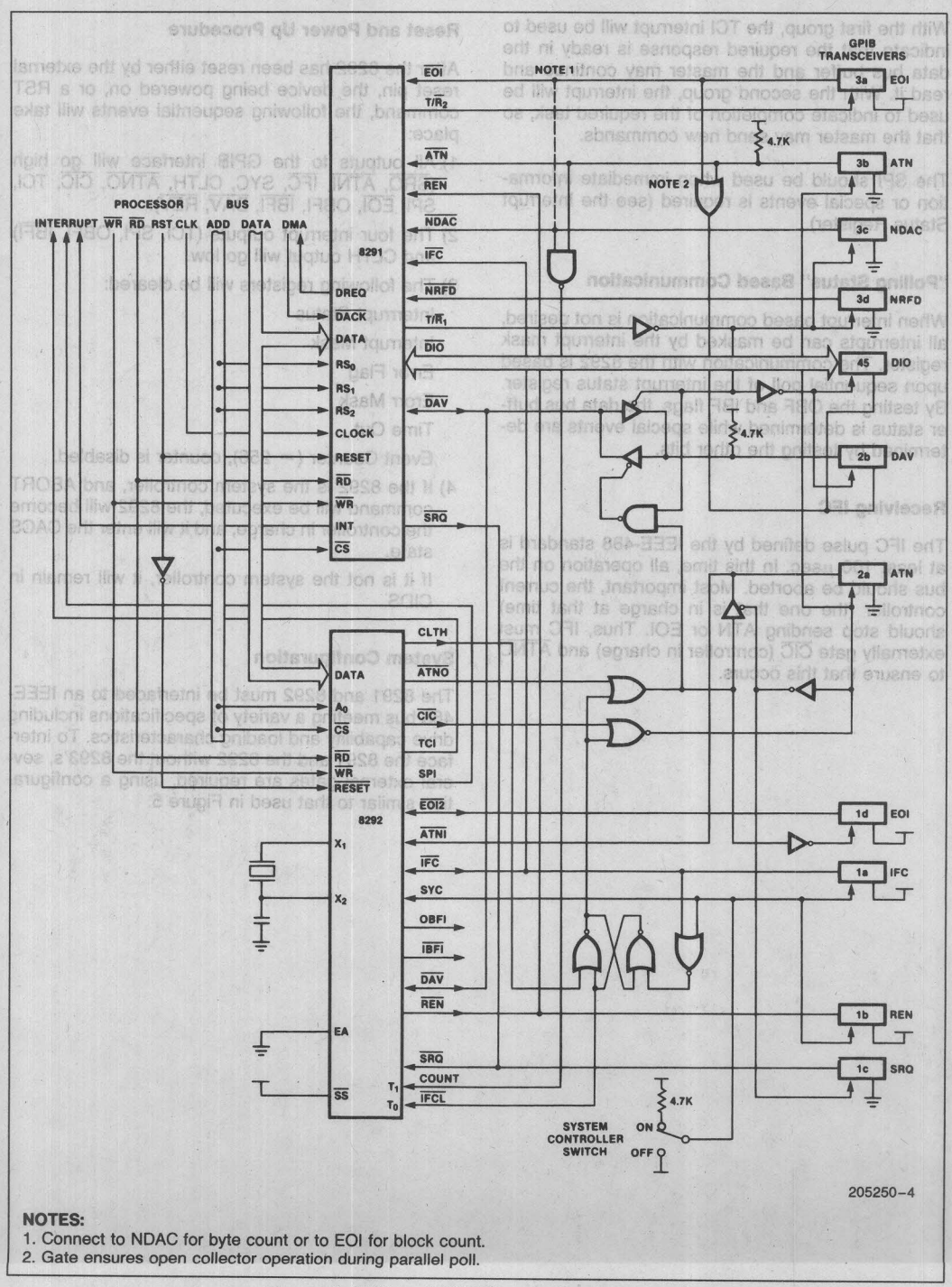
After the 8292 has been reset either by the external reset pin, the device being powered on, or a RST command, the following sequential events will take place:

1) All outputs to the GPIB interface will go high ($\overline{SRQ}$, $\overline{ATNI}$, $\overline{IFC}$, SYC, CLTH, $\overline{ATNO}$, $\overline{CIC}$, TCI, SPI, $\overline{EOI}$, OBFI, $\overline{IBFI}$, $\overline{DAV}$, $\overline{REV}$).
2) The four interrupt outputs (TCI, SPI, OBFI, $\overline{IBFI}$) and CLTH output will go low.
3) The following registers will be cleared:
   - Interrrupt Status
   - Interrupt Mask
   - Error Flag
   - Erorr Mask
   - Time Out
   - Event Counter (= 256), counter is disabled.
4) If the 8292 is the system controller, and ABORT command will be executed, the 8292 will become the controller in charge, and it will enter the CACS state.

   If it is not the system controller, it will remain in CIDS.

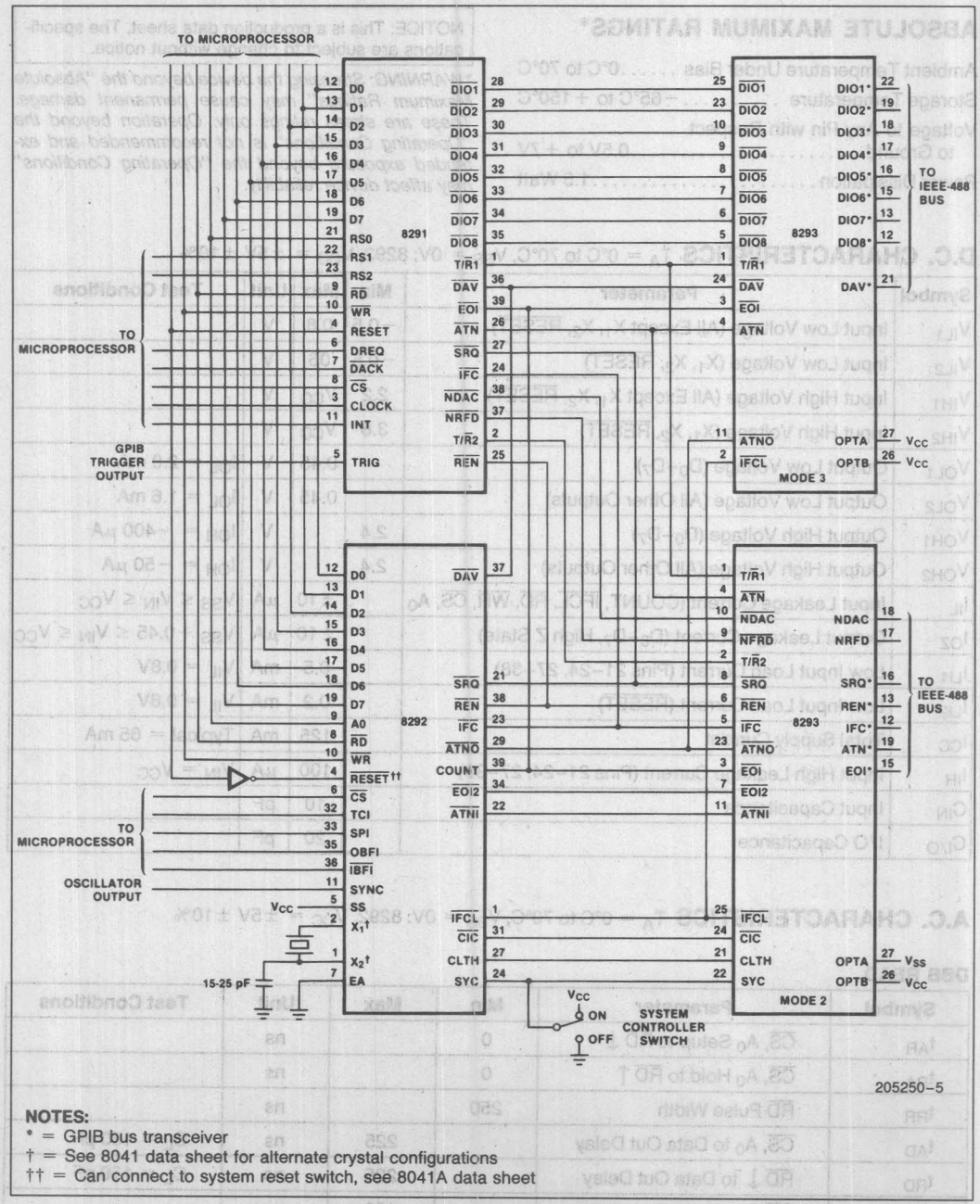
System Configuration

The 8291 and 8292 must be interfaced to an IEEE-488 bus meeting a variety of specifications including drive capability and loading characteristics. To interface the 8291 and the 8292 without the 8293's, several external gates are required, using a configuration similar to that used in Figure 5.



NOTES:
1. Connect to NDAC for byte count or to EOI for block count.
2. Gate ensures open collector operation during parallel poll.

Figure 4. 8291 and 8292 System Configuration



NOTES:
* = GPIB bus transceiver
† = See 8041 data sheet for alternate crystal configurations
†† = Can connect to system reset switch, see 8041A data sheet

Figure 5. 8291, 8292, and 8293 System Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage to Any Pin with Respect to Ground.....0.5V to +7V
Power Dissipation.....1.5 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{SS} = 0V: 8292, V_{CC} = ±5V ±10%

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL1}	Input Low Voltage (All Except X_1, X_2, $\overline{RESET}$)	−0.5	0.8	V	
V_{IL2}	Input Low Voltage (X_1, X_2, $\overline{RESET}$)	−0.5	06	V	
V_{IH1}	Input High Voltage (All Except X_1, X_2, $\overline{RESET}$)	2.2	V_{CC}	V	
V_{IH2}	Input High Voltage (X_1, X_2, $\overline{RESET}$)	3.8	V_{CC}	V	
V_{OL1}	Output Low Voltage (D_0–D_7)		0.45	V	I_{OL} = 2.0 mA
V_{OL2}	Output Low Voltage (All Other Outputs)		0.45	V	I_{OL} = 1.6 mA
V_{OH1}	Output High Voltage (D_0–D_7)	2.4		V	I_{OH} = −400 μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4		V	I_{OH} = −50 μA
I_{IL}	Input Leakage Current (COUNT, $\overline{IFCL}$, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0		±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OZ}	Output Leakage Current (D_0–D_7, High Z State)		±10	μA	V_{SS} +0.45 ≤ $V_{IN} \le V_{CC}$
I_{LI1}	Low Input Load Current (Pins 21–24, 27–38)		0.5	mA	V_{IL} = 0.8V
I_{LI2}	Low Input Load Current ($\overline{RESET}$)		0.2	mA	V_{IL} = 0.8V
I_{CC}	Total Supply Current		125	mA	Typical = 65 mA
I_{IH}	Input High Leakage Current (Pins 21–24, 27–38)		100	μA	$V_{IN} = V_{CC}$
C_{IN}	Input Capacitance		10	pF	
$C_{I/O}$	I/O Capacitance		20	pF	

A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{SS} = 0V: 8292, V_{CC} = ±5V ±10%

DBB READ

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$ ↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold to $\overline{RD}$ ↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	C_L = 150 pF
t_{RD}	$\overline{RD}$ ↓ to Data Out Delay		225	ns	C_L = 150 pF
t_{DF}	$\overline{RD}$ ↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time	2.5	15	μs	

DBB WRITE

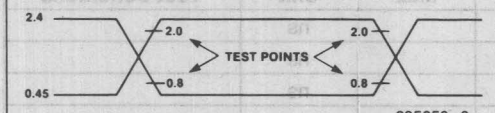
Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	CS, A_0 Setup to WR ↓	0		ns	
t_{WA}	CS, A_0 Hold after WR ↑	0		ns	
t_{WW}	WR Pulse Width	250		ns	
t_{DW}	Data Setup to WR ↑	150		ns	
t_{WD}	Data Hold after WR ↓	0		ns	

COMMAND TIMINGS(1, 3)

Code	Name	Execution Time	$\overline{IBFI}$ ↑	TCI(2)	SPI	$\overline{ATNO}$	$\overline{CIC}$	$\overline{IFC}$	$\overline{REN}$	$\overline{EOI}$	$\overline{DAV}$	Comments
E1	WTOUT	63	24									
E2	WEVC	63	24									
E3	REVC	71	24	51								
E4	RERF	67	24	47								
E5	RINM	69	24	49								
E6	RCST	97	24	77								
E7	RBST	92	24	72								
E8												
E9	RTOUT	69	24	49								
EA	RERM	69	24	49								
F0	SPCNI	53	24									Count Stops after 39
F1	GIOL	88	24	70		↑61	↑61					
F2	RST	94	24		↓52							Not System Controller
F2	RST	214	24	192	↓52	↓179	↓174	↓101				System Controller
F3	RSTI	61	24									
F4	GSEC	125	24	107		↑98						
F5	EXPP	75	24						↓53 ↑59	↓55 ↑57		
F6	GTSB	118	24	100		↑91						
F7	SLOC	73	24	55				↑46				
F8	SREM	91	24	73				↓64				
F9	ABORT	155	24	133		↓120	↓115	↓42				
FA	TCNTR	108	24	86		↓71	↓68					
FC	TCAS	92	24	67		↓55						
FD	TCSY	115	24	91		↓80						
FE	STCNI	59	24									Starts Count after 43
PIN	RESET	29	—	↓7	↓7							Not System Controller
X	IACK	116	—		↓73 ↑98							If Interrupt Pending

NOTES:
1. All times are multiples of t_{CY} from the 8041A command interrupt.
2. TCI clears after 7 t_{CY} on all commands.
3. ↑ indicates a level transition from low to high, ↓ indicates a high to low transition.

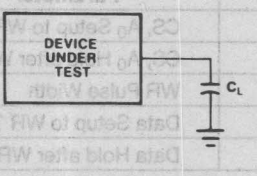
A.C. TESTING INPUT, OUTPUT WAVEFORM



205250-6

A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

A.C. TESTING LOAD CIRCUIT

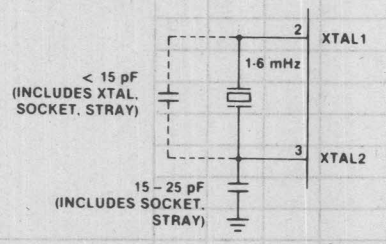


205250-7

C_L Include Jig Capacitance.

CLOCK DRIVER CIRCUITS

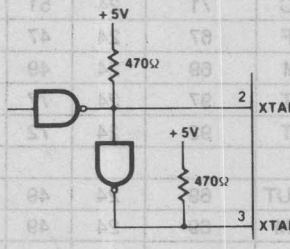
CRYSTAL OSCILLATOR MODE



205250-8

NOTE:
Crystal series resistance should be <75Ω at 6 MHz; <180Ω at 3.6 MHz.

DRIVING FROM EXTERNAL SOURCE

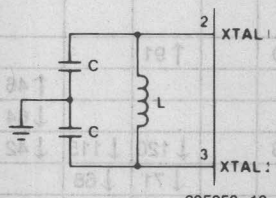


205250-9

NOTE:
Both XTAL1 and XTAL2 should be driven. Resistors to V_{CC} are needed to ensure V_{IH} = 2.8V if TTL circuitry is used.

LC OSCILLATOR MODE

L	C	NOMINAL f
45 μH	20 pF	5.2 MHz
120 μH	20 pF	3.2 MHz



205250-10

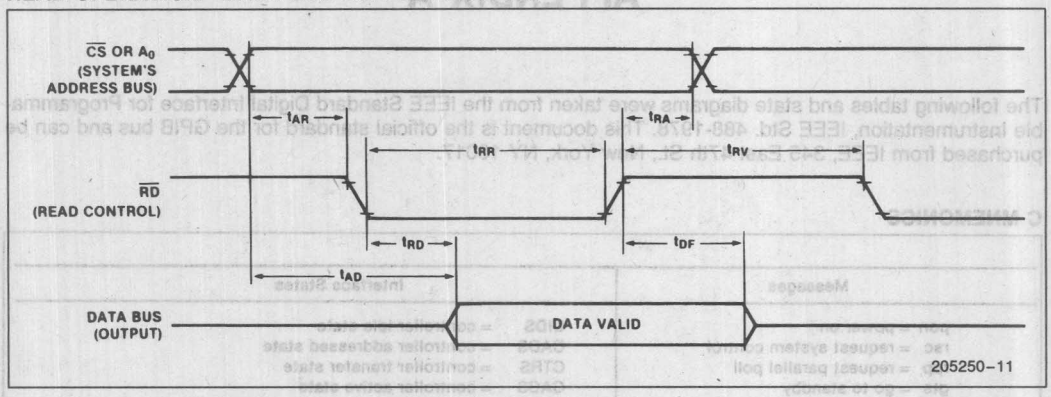
$$f = \frac{1}{2\pi\sqrt{LC'}}$$

$$C' = \frac{C + 3Cpp}{2}$$

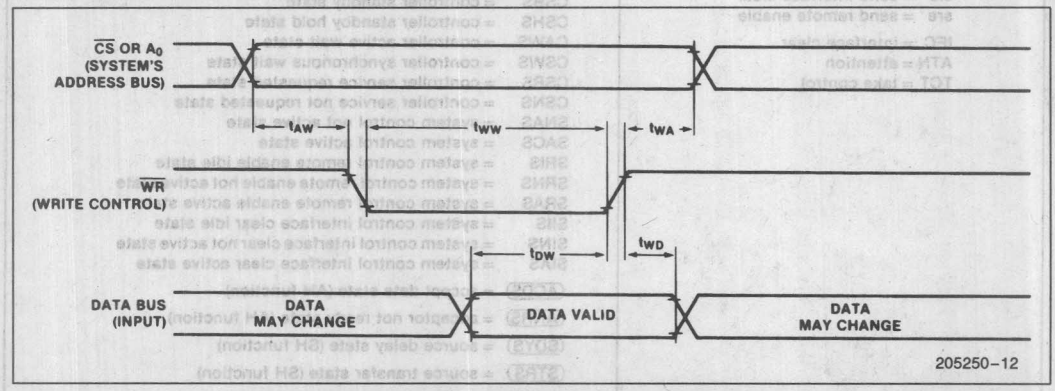
NOTES:
1. Cpp ≈ 5–10 pF pin-to-pin capacitance
2. Each C should be approximately 20 pF, including stray capacitance.

WAVEFORMS

READ OPERATION—DATA BUS BUFFER REGISTER



WRITE OPERATION—DATA BUS BUFFER REGISTER



3

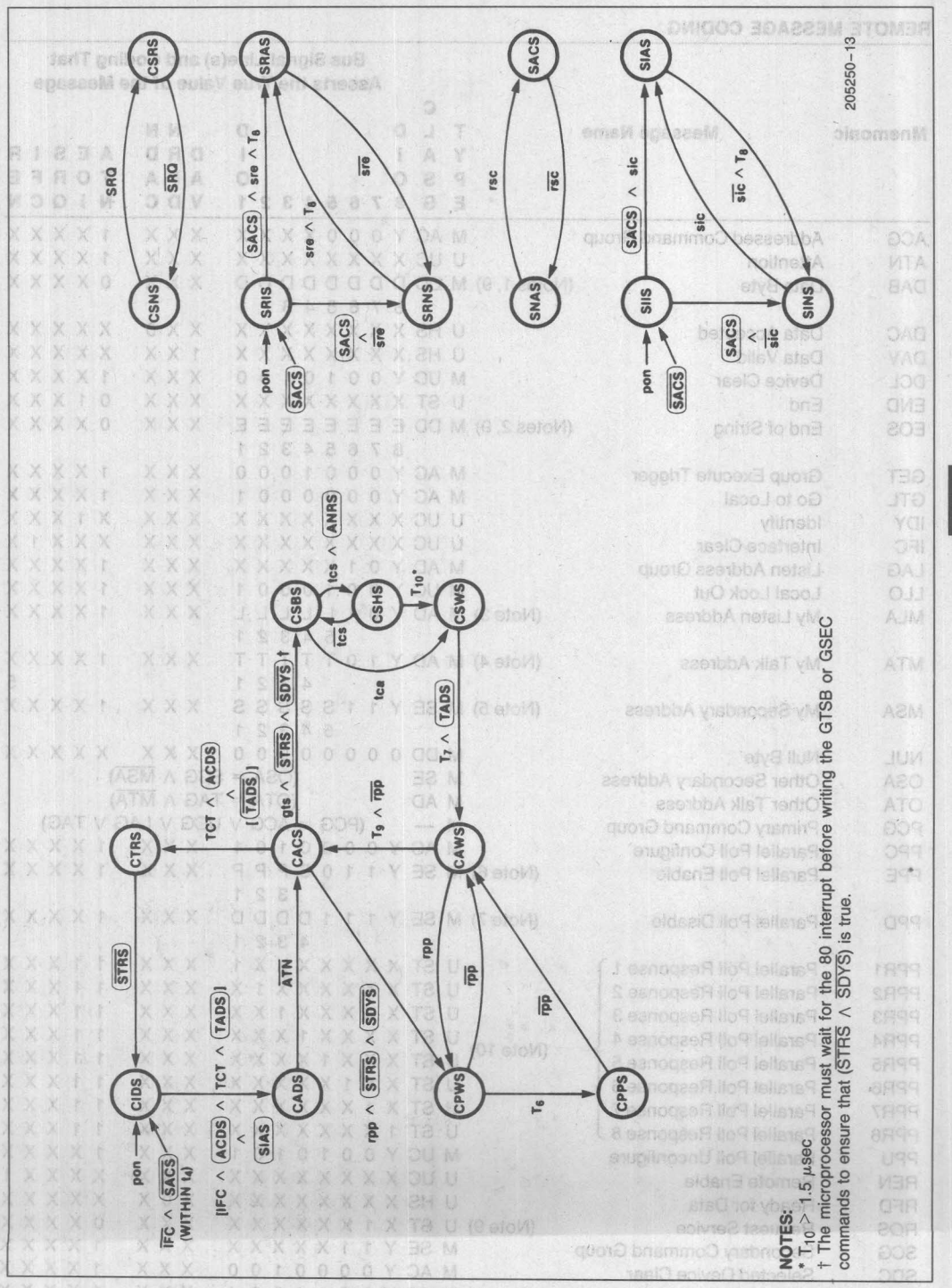
APPENDIX A

The following tables and state diagrams were taken from the IEEE Standard Digital Interface for Programmable Instrumentation, IEEE Std. 488-1978. This document is the official standard for the GPIB bus and can be purchased from IEEE, 345 East 47th St., New York, NY 10017.

C MNEMONICS

Messages	Interface States
pon = power on	CIDS = controller idle state
rsc = request system control	CADS = controller addressed state
rpp = request parallel poll	CTRS = controller transfer state
gts = go to standby	CACS = controller active state
tca = take control asynchronously	CPWS = controller parallel poll wait state
tcs = take control synchronously	CPPS = controller parallel poll state
sic = send interface clear	CSBS = controller standby state
sre = send remote enable	CSHS = controller standby hold state
IFC = interface clear	CAWS = controller active wait state
ATN = attention	CSWS = controller synchronous wait state
TCT = take control	CSRS = controller service requested state
	CSNS = controller service not requested state
	SNAS = system control not active state
	SACS = system control active state
	SRIS = system control remote enable idle state
	SRNS = system control remote enable not active state
	SRAS = system control remote enable active state
	SIIS = system control interface clear idle state
	SINS = system control interface clear not active state
	SIAS = system control interface clear active state
	(ACDS) = accept data state (AH function)
	(ANRS) = acceptor not ready state (AH function)
	(SDYS) = source delay state (SH function)
	(STRS) = source transfer state (SH function)
	(TADS) = talker addressed state (T function)

205250-14



NOTES:
* $T_{10} > 1.5\ \mu sec$
† The microprocessor must wait for the 80 interrupt before writing the GTSB or GSEC commands to ensure that (STRS ∧ SDYS) is true.

Figure A-1. C State Diagram

REMOTE MESSAGE CODING

Mnemonic	Message Name		Bus Signal Line(s) and Coding That Asserts the True Value of the Message																				
			TYPE	CLASS	DIO8	7	6	5	4	3	2	DIO1	DAV	NRFD	NDAC	ATN	EOI	SRQ	IFC	REN
ACG	Addressed Command Group		M	AC	Y	0	0	0	X	X	X	X	X	X	X	1	X	X	X	X
ATN	Attention		U	UC	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X
DAB	Data Byte	(Notes 1, 9)	M	DD	D8	D7	D6	D5	D4	D3	D2	D1	X	X	X	0	X	X	X	X
DAC	Data Accepted		U	HS	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X
DAV	Data Valid		U	HS	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X
DCL	Device Clear		M	UC	Y	0	0	1	0	1	0	0	X	X	X	1	X	X	X	X
END	End		U	ST	X	X	X	X	X	X	X	X	X	X	X	0	1	X	X	X
EOS	End of String	(Notes 2, 9)	M	DD	E8	E7	E6	E5	E4	E3	E2	E1	X	X	X	0	X	X	X	X
GET	Group Execute Trigger		M	AC	Y	0	0	0	1	0	0	0	X	X	X	1	X	X	X	X
GTL	Go to Local		M	AC	Y	0	0	0	0	0	0	1	X	X	X	1	X	X	X	X
IDY	Identify		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X
IFC	Interface Clear		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X
LAG	Listen Address Group		M	AD	Y	0	1	X	X	X	X	X	X	X	X	1	X	X	X	X
LLO	Local Lock Out		M	UC	Y	0	0	1	0	0	0	1	X	X	X	1	X	X	X	X
MLA	My Listen Address	(Note 3)	M	AD	Y	0	1	L5	L4	L3	L2	L1	X	X	X	1	X	X	X	X
MTA	My Talk Address	(Note 4)	M	AD	Y	1	0	T4	T3	T2	T1	T5	X	X	X	1	X	X	X	X
MSA	My Secondary Address	(Note 5)	M	SE	Y	1	1	S5	S4	S3	S2	S1	X	X	X	1	X	X	X	X
NUL	Null Byte		M	DD	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X
OSA	Other Secondary Address		M	SE	(OSA = SCG ∧ $\overline{MSA}$)															
OTA	Other Talk Address		M	AD	(OTA = TAG ∧ $\overline{MTA}$)															
PCG	Primary Command Group		M	—	(PCG = ACG ∨ UCG ∨ LAG ∨ TAG)															
PPC	Parallel Poll Configure		M	AC	Y	0	0	0	0	1	0	1	X	X	X	1	X	X	X	X
PPE	Parallel Poll Enable	(Note 6)	M	SE	Y	1	1	0	S	P3	P2	P1	X	X	X	1	X	X	X	X
PPD	Parallel Poll Disable	(Note 7)	M	SE	Y	1	1	1	D4	D3	D2	D1	X	X	X	1	X	X	X	X
PPR1	Parallel Poll Response 1	(Note 10)	U	ST	X	X	X	X	X	X	X	1	X	X	X	1	1	X	X	X
PPR2	Parallel Poll Response 2		U	ST	X	X	X	X	X	X	1	X	X	X	X	1	1	X	X	X
PPR3	Parallel Poll Response 3		U	ST	X	X	X	X	X	1	X	X	X	X	X	1	1	X	X	X
PPR4	Parallel Poll Response 4		U	ST	X	X	X	X	1	X	X	X	X	X	X	1	1	X	X	X
PPR5	Parallel Poll Response 5		U	ST	X	X	X	1	X	X	X	X	X	X	X	1	1	X	X	X
PPR6	Parallel Poll Response 6		U	ST	X	X	1	X	X	X	X	X	X	X	X	1	1	X	X	X
PPR7	Parallel Poll Response 7		U	ST	X	1	X	X	X	X	X	X	X	X	X	1	1	X	X	X
PPR8	Parallel Poll Response 8		U	ST	1	X	X	X	X	X	X	X	X	X	X	1	1	X	X	X
PPU	Parallel Poll Unconfigure		M	UC	Y	0	0	1	0	1	0	1	X	X	X	1	X	X	X	X
REN	Remote Enable		U	UC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1
RFD	Ready for Data		U	HS	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X
RQS	Request Service	(Note 9)	U	ST	X	1	X	X	X	X	X	X	X	X	X	0	X	X	X	X
SCG	Secondary Command Group		M	SE	Y	1	1	X	X	X	X	X	X	X	X	1	X	X	X	X
SDC	Selected Device Clear		M	AC	Y	0	0	0	0	1	0	0	X	X	X	1	X	X	X	X
SPD	Serial Poll Disable		M	UC	Y	0	0	1	1	0	0	1	X	X	X	1	X	X	X	X

REMOTE MESSAGE CODING (Continued)

Mnemonic	Message Name		Bus Signal Line(s) and Coding That Asserts the True Value of the Message																				
			TYPE	CLAS	DIO8	7	6	5	4	3	2	1	DIO	DAV	NRFD	NDAC	ATN	EOI	SRQ	IFC	REN
SPE	Serial Poll Enable		M	UC	Y	0	0	1	1	0	0	0		X	X	X	1	X	X	X	X
SRQ	Service Request		U	ST	X	X	X	X	X	X	X	X		X	X	X	X	X	1	X	X
STB	Status Byte	(Notes 8, 9)	M	ST	S8	X	S6	S5	S4	S3	S2	S1		X	X	X	0	X	X	X	X
TCT	Take Control		M	AC	Y	0	0	0	1	0	0	1		X	X	X	1	X	X	X	X
TAG	Talk Address Group		M	AD	Y	1	0	X	X	X	X	X		X	X	X	1	X	X	X	X
UCG	Universal Command Group		M	UC	Y	0	0	1	X	X	X	X		X	X	X	1	X	X	X	X
UNL	Unlisten		M	1D	Y	0	1	1	1	1	1	1		X	X	X	1	X	X	X	X
UNT	Untalk	(Note 11)	M	1D	Y	1	0	1	1	1	1	1		X	X	X	1	X	X	X	X

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:
1. D1–D8 specify the device dependent data bits.
2. E1–E8 specify the device dependent code used to indicate the EOS message.
3. L1–L5 specify the device dependent bits of the device's listen address.
4. T1–T5 specify the device dependent bits of the device's talk address.
5. S1–S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.
   Response = $\overline{S \oplus ist}$

P1–P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

7. D1–D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1–S6, S8 specify the device dependent status (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use.

8294A
DATA ENCRYPTION/DECRYPTION UNIT

- **Certified by National Bureau of Standards**
- **400 Byte/Sec Data Conversion Rate**
- **64-Bit Data Encryption Using 56-Bit Key**
- **DMA Interface**
- **3 Interrupt Outputs to Aid in Loading and Unloading Data**
- **7-Bit User Output Port**
- **Single 5V ± 10% Power Supply**
- **Fully Compatible with iAPX-86, 88, MCS-85™, MCS-80™, MCS-51™, and MCS-48™ Processors**
- **Implements Federal Information Processing Data Encryption Standard**
- **Encrypt and Decrypt Modes Available**

The Intel® 8294A Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294A; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294A in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 400 bytes/second. The 8294A also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294A implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

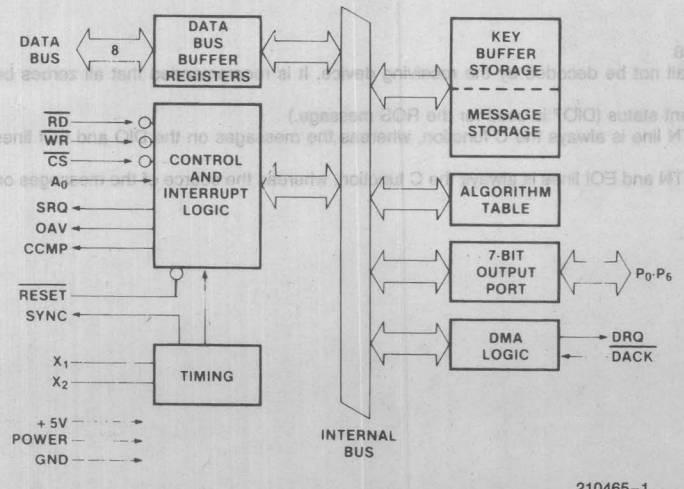


Figure 1. Block Diagram

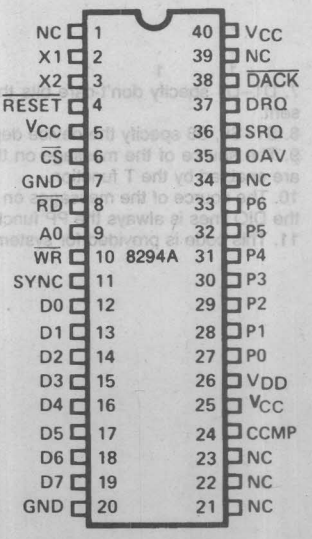


Figure 2. Pin Configuration

November 1986
Order Number: 210465-004

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
NC	1		**NO CONNECTION.**
X1 X2	2 3		**CRYSTAL:** Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
$\overline{RESET}$	4	I	**RESET:** A low signal to this pin resets the 8294A.
V_{CC}	5		**POWER:** Tied high.
$\overline{CS}$	6	I	**CHIP SELECT:** A low signal to this pin enables reading and writing to the 8294A.
GND	7		**GROUND:** This pin must be tied to ground.
$\overline{RD}$	8	I	**READ:** An active low read strobe at this pin enables the CPU to read data and status from the internal DEU registers.
A_0	9	I	**ADDRESS:** Address input used by the CPU to select DEU registers during read and write operations.
$\overline{WR}$	10	I	**WRITE:** An active low write strobe at this pin enables the CPU to send data and commands to the DEU.
SYNC	11	O	**SYNC:** High frequency (Clock ÷ 15) output. Can be used as a strobe for external circuitry.
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7	12 13 14 15 16 17 18 19	I/O	**DATA BUS:** Three-state, bi-directional data bus lines used to transfer data between the CPU and the 8294A.
GND	20		**GROUND:** This pin must be tied to ground.
V_{CC}	40		**POWER:** +5V power input: +5V ±10%.
NC	39		**NO CONNECTION.**
$\overline{DACK}$	38	I	**DMA ACKNOWLEDGE:** Input signal from the 8257 DMA Controller acknowledging that the requested DMA cycle has been granted.
DRQ	37	O	**DMA REQUEST:** Output signal to the 8257 DMA Controller requesting a DMA cycle.
SRQ	36	O	**SERVICE REQUEST:** Interrupt to the CPU indicating that the 8294A is awaiting data or commands at the input buffer. SRQ = 1 implies IBF = 0.
OAV	35	O	**OUTPUT AVAILABLE:** Interrupt to the CPU indicating that the 8294A has data or status available in its output buffer, OAV = 1 implies OBF = 1.
NC	34		**NO CONNECTION.**

3

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
P6 P5 P4 P3 P2 P1 P0	33 32 31 30 29 28 27	O	**OUTPUT PORT:** User output port lines. Output lines available to the user via a CPU command which can asset selected port lines. These lines have nothing to do with the encryption function. At power-on, each line is in a 1 state.
V_{DD}	26		**POWER:** +5V power input. (+5V ±10%) Low power standby pin.
V_{CC}	25		**POWER:** Tied high.
CCMP	24	O	**CONVERSION COMPLETE:** Interrupt to the CPU indicating that the encryption/decryption of an 8-byte block is complete.
NC	23		**NO CONNECTION.**
NC	22		**NO CONNECTION.**
NC	21		**NO CONNECTION.**

FUNCTIONAL DESCRIPTION

OPERATION

The data conversion sequence is as follows:

1) A Set Mode command is given, enabling the desired interrupt outputs.
2) An Enter New Key command is issued, followed by 8 data inputs which are retained by the DEU for encryption/decryption. Each byte must have odd parity.
3) An Encrypt Data or Decrypt Data command sets the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a Decrypt Data command could be issued to change the DEU from encrypt mode to decrypt mode without changing either the key or the interrupt outputs enabled.

INTERNAL DEU REGISTERS

Four internal registers are addressable by the master processor: 2 for input, and 2 for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	A_0	Register
1	0	0	0	Data Input Buffer
0	1	0	0	Data Output Buffer
1	0	0	1	Command Input Buffer
0	1	0	1	Status Output Buffer
X	X	1	X	Don't Care

The functions of each of these registers are described below.

Data Input Buffer—Data written to this register is interpreted in one of three ways, depending on the preceding command sequence.

1) Part of a key.
2) Data to be encrypted or decrypted.
3) A DMA block count.

Data Output Buffer—Data read from this register is the output of the encryption/decryption operation.

Command Input Buffer—Commands to the DEU are written into this register. (See command summary below.)

Status Output Buffer—DEU status is available in this register at all times. It is used by the processor for poll-driven command and data transfer operations.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	X	X	X	KPE	CF	DEC	IBF	OBF

OBF Output Buffer Full; OBF = 1 indicates that output from the encryption/decryption function is available in the Data Output Buffer. It is reset when the data is read.

IBF Input Buffer Full; A write to the Data Input Buffer or to the Command Input Buffer sets IBF = 1. The DEU resets this flag when it has accepted the input byte. Nothing should be written when IBF = 1.

DEC Decrypt; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the decrypt mode. DEC = 0 implies the encrypt mode.

After 8294A has accepted a 'Decrypt Data' or 'Encrypt Data' command, 11 cycles are required to update the DEC bit.

CF Completion Flag; This flag may be used to indicate any or all of three events in the data transfer protocol.

1) It may be used in lieu of a counter in the processor routine to flag the end of an 8-byte transfer.
2) It must be used to indicate the validity of the KPE flag.
3) It may be used in lieu of the CCMP interrupt to indicate the completion of a DMA operation.

KPE Key Parity Error; After a new key has been entered, the DEU uses this flag in conjunction with the CF flag to indicate correct or incorrect parity.

COMMAND SUMMARY

1 — Enter New Key

OP CODE:	0	1	0	0	0	0	0	0
	MSB							LSB

This command is followed by 8 data byte inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data. These data bytes must have odd parity represented by the LSB.

2 — Encrypt Data

OP CODE:	0	0	1	1	0	0	0	0
	MSB							LSB

This command puts the 8294A into the encrypt mode.

3 — Decrypt Data

OP CODE:	0	0	1	0	0	0	0	0
	MSB							LSB

This command puts the 8294A into the decrypt mode.

4 — Set Mode

OP CODE:	0	0	0	0	A	B	C	D
	MSB							LSB

where:

A is the OAV (Output Available) interrupt enable
B is the SRQ (Service Request) interrupt enable
C is the DMA (Direct Memory Access) transfer enable
D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A, B = 1). Following the command in which bit C, the DMA bit, is set, the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

5 — Write to Output Port

OP CODE:	1	P_6	P_5	P_4	P_3	P_2	P_1	P_0
	MSB							LSB

This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port. The initial output is 1111111. Use of this port is independent of the encryption/decryption function.

PROCESSOR/DEU INTERFACE PROTOCOL

ENTERING A NEW KEY

The timing sequence for entering a new key is shown in Figure 3. A flowchart showing the CPU software to accommodate this sequence is given in Figure 4.

After the Enter New Key command is issued, 8 data bytes representing the new key are written to the data input buffer (most significant byte first). After the eighth byte is entered into the DEU, CF goes true (CF = 1). The CF bit goes false again when KPE is valid. The CPU can then check the KPE flag. If KPE = 1, a parity error has been detected and the DEU has not accepted the key. Each byte is checked for odd parity, where the parity bit is the LSB of each byte.

210465-3

Figure 3. Entering a New Key

Since CF = 1 only for a short period of time after the last byte is accepted, the CPU which polls the CF flag might miss detecting CF = 1 momentarily. Thus, a counter should be used, as in Figure 4, to flag the end of the new key entry. Then CF is used to indicate a valid KPE flag.

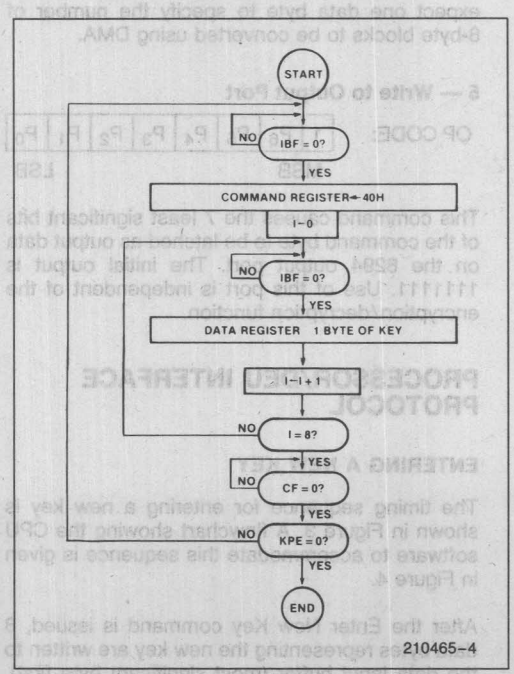


Figure 4. Flowchart for Entering a New Key

and remains false, after the first read. After 8 bytes have been read back by the CPU, CF goes false (CF = 0). Thus, the CPU may test for CF = 0 to terminate the read mode. Also, the CCMP interrupt may be used to initiate a service routine which performs the next series of 8 data reads and 8 data writes.

Figure 6 offers two flowcharts outlining the alternative means of implementing the data conversion protocol. Either the CF flag or a software counter may be used to end the read and write modes.

SRQ = 1 implies IBF = 0, OAV = 1 implies OBF = 1. This allows interrupt routines to do data transfers without checking status first. However, the OAV service routine must detect and flag the end of a data conversion.

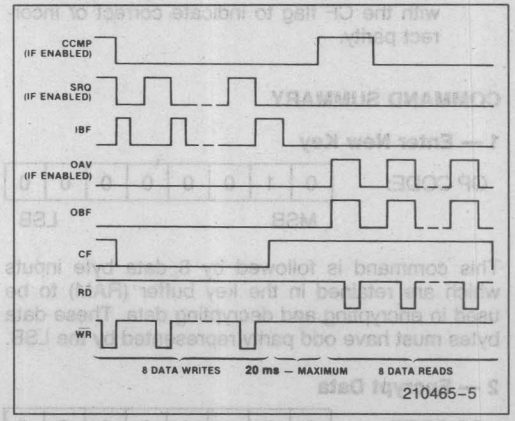
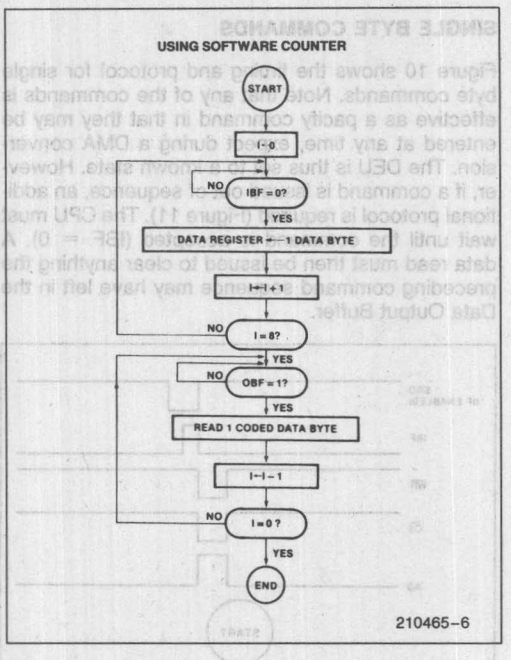


Figure 5. Encrypting/Decrypting Data



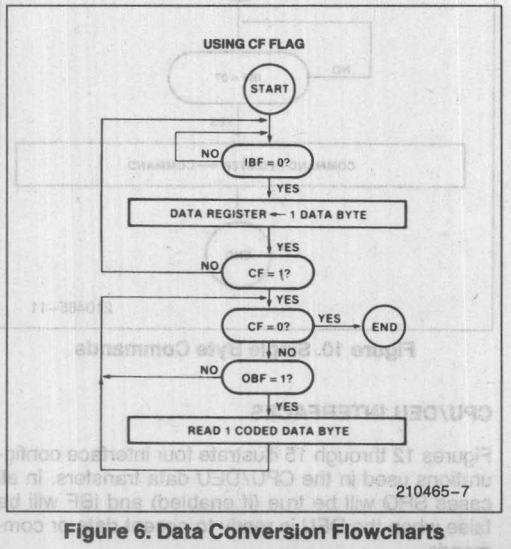


Figure 6. Data Conversion Flowcharts

USING DMA

The timing sequence for data conversions using DMA is shown in Figure 7. This sequence can be better understood when considered in conjunction with the hardware DMA interface in Figure 8. Note

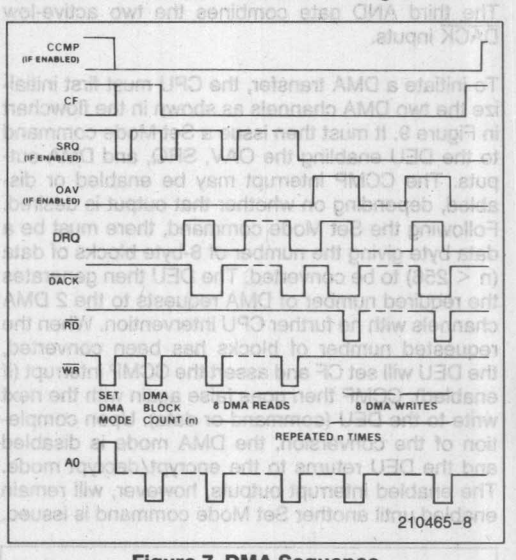


Figure 7. DMA Sequence

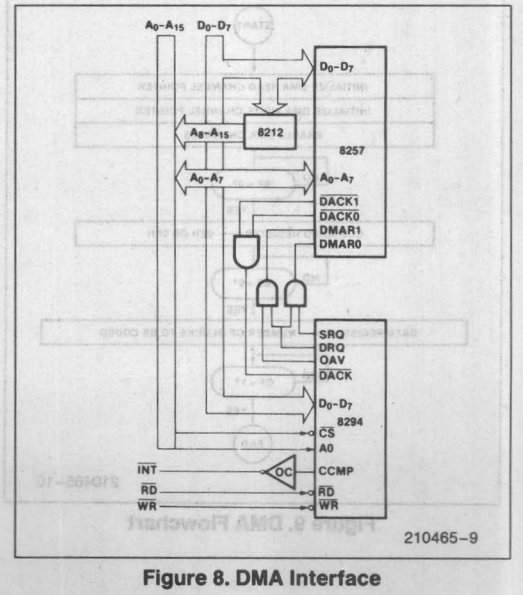


Figure 8. DMA Interface

that the use of the DMA feature requires 3 external AND gates and 2 DMA channels (one for input, one for output). Since the DEU has only one DMA request pin, the SRQ and OAV outputs are used in conjunction with two of the AND gates to create separate DMA request outputs for the 2 DMA channels. The third AND gate combines the two active-low $\overline{DACK}$ inputs.

To initiate a DMA transfer, the CPU must first initialize the two DMA channels as shown in the flowchart in Figure 9. It must then issue a Set Mode command to the DEU enabling the OAV, SRQ, and DMA outputs. The CCMP interrupt may be enabled or disabled, depending on whether that output is desired. Following the Set Mode command, there must be a data byte giving the number of 8-byte blocks of data ($n < 256$) to be converted. The DEU then generates the required number of DMA requests to the 2 DMA channels with no further CPU intervention. When the requested number of blocks has been converted, the DEU will set CF and assert the CCMP interrupt (if enabled). CCMP then goes false again with the next write to the DEU (command or data). Upon completion of the conversion, the DMA mode is disabled and the DEU returns to the encrypt/decrypt mode. The enabled interrupt outputs, however, will remain enabled until another Set Mode command is issued.

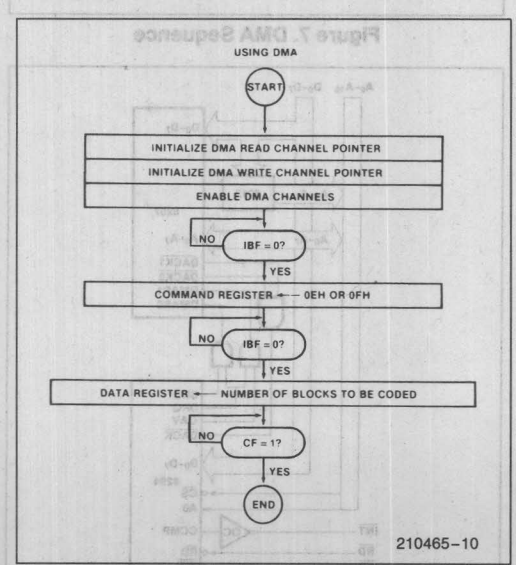


Figure 9. DMA Flowchart

SINGLE BYTE COMMANDS

Figure 10 shows the timing and protocol for single byte commands. Note that any of the commands is effective as a pacify command in that they may be entered at any time, expect during a DMA conversion. The DEU is thus set to a known state. However, if a command is issued out of sequence, an additional protocol is required (Figure 11). The CPU must wait until the command is accepted (IBF = 0). A data read must then be issued to clear anything the preceding command sequence may have left in the Data Output Buffer.

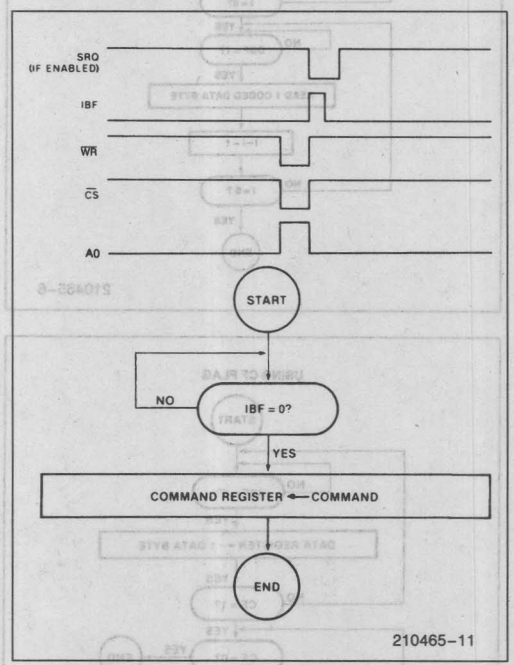


Figure 10. Single Byte Commands

CPU/DEU INTERFACES

Figures 12 through 15 illustrate four interface configurations used in the CPU/DEU data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.

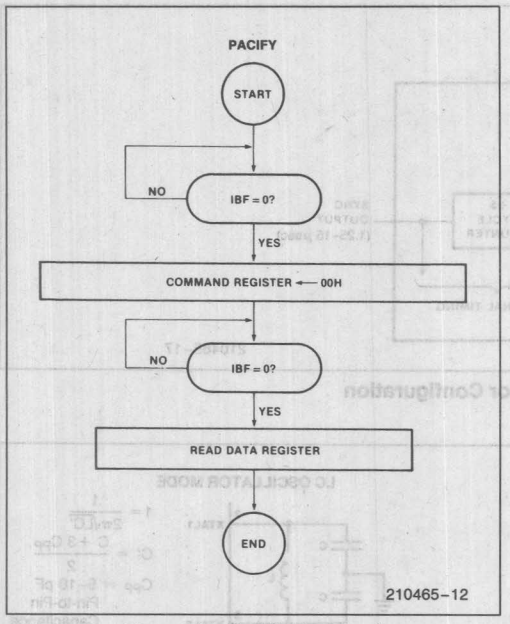


Figure 11. Pacify Protocol

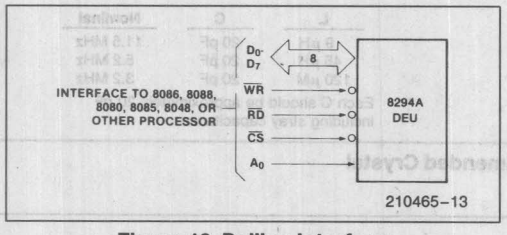


Figure 12. Polling Interface

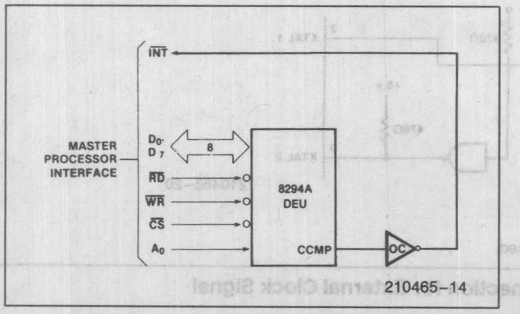


Figure 13. Single Interrupt Interface

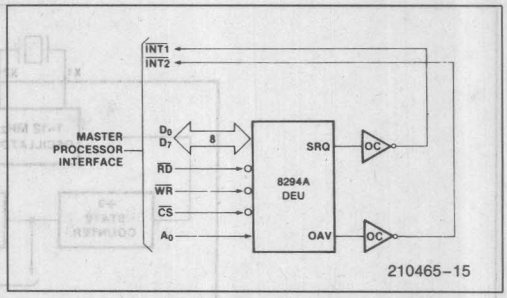
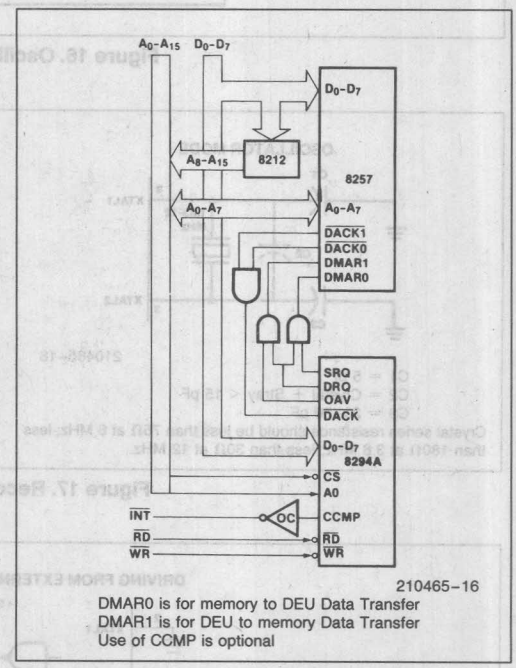


Figure 14. Dual Interrupt Interface



DMAR0 is for memory to DEU Data Transfer
DMAR1 is for DEU to memory Data Transfer
Use of CCMP is optional

Figure 15. DMA Interface

OSCILLATING AND TIMING CIRCUITS

The 8294A's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 16.

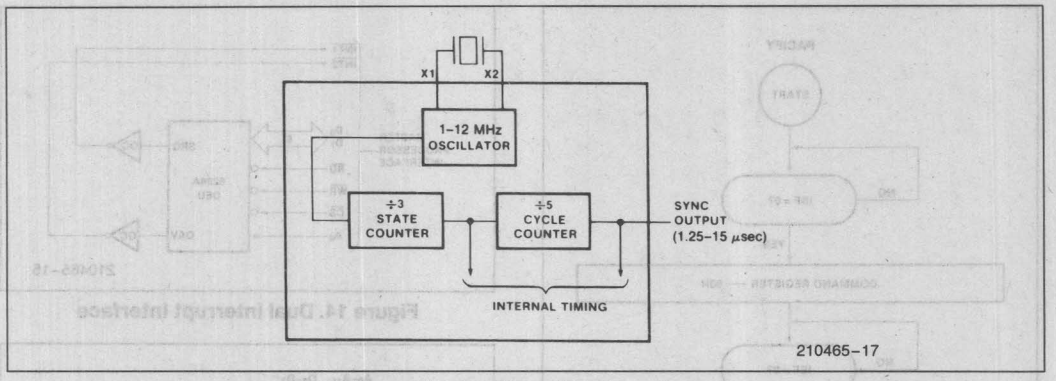


Figure 16. Oscillator Configuration

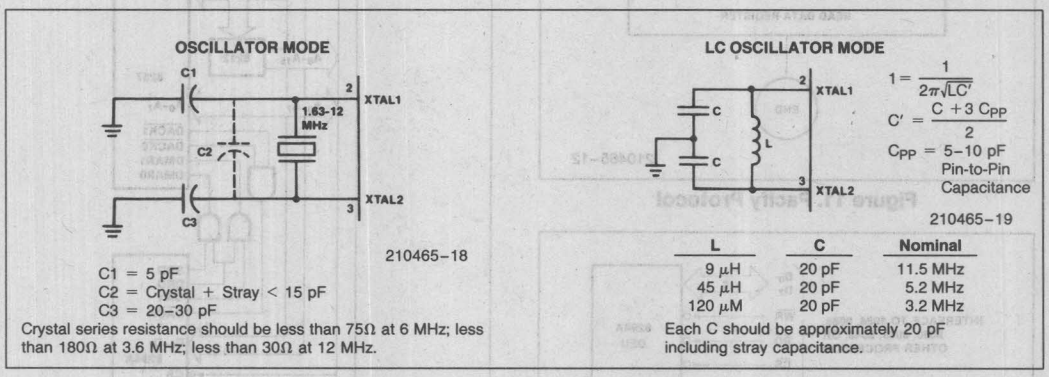


Figure 17. Recommended Crystal

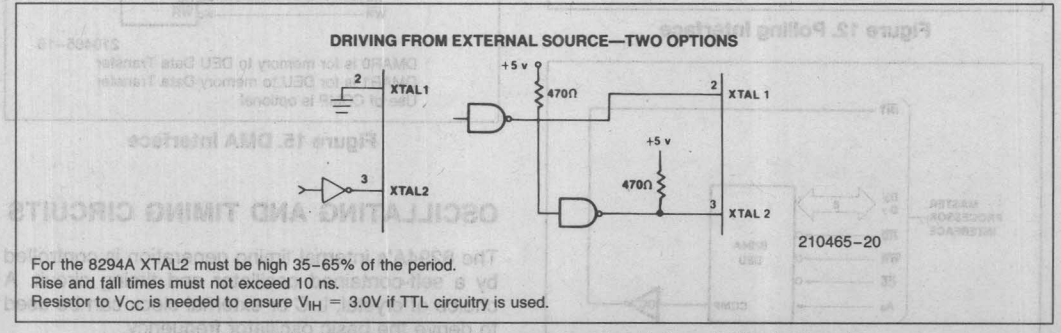


Figure 18. Recommended Connection for External Clock Signal

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With Respect to Ground . . . . . −0.5V to +7V
Power Dissipation . . . . . 1.5 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V_{IL}	Input Low Voltage (All Except X_1, X_2, RESET)	−0.5		0.8	V	
V_{IL1}	Input Low Voltage (X_1, X_2, RESET	−0.5		0.6	V	
V_{IH}	Input High Voltage (All Except X_1, RESET)	2.0		V_{CC}	V	
V_{IH1}	Input High Voltage (X_1, RESET	3.5		V_{CC}	V	
V_{IH2}	Input High Voltage (X_2)	2.2		V_{CC}	V	
V_{OL}	Output Low Voltage (D_0–D_7)			0.45	V	I_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage (All Other Outputs)			0.45	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage (D_0–D_7)	2.4			V	I_{OH} = −400 μA
V_{OH1}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = −50 μA
I_{IL}	Input Leakage Current (RD, WR, CS, A_0)			±10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OFL}	Output Leakage Current (D_0–D_7, High Z State)			±10	μA	$V_{SS} + 0.45 \le V_{OUT} \le V_{CC}$
I_{DD}	V_{DD} Supply Current		5	20	mA	
$I_{DD} + I_{CC}$	Total Supply Current		60	135	mA	
I_{LI}	Low Input Load Current (Pins 24, 27–38)			0.3	mA	V_{IL} = 0.8V
I_{LI1}	Low Input Load Current (RESET)			0.2	mA	V_{IL} = 0.8V
I_{IH}	Input High Leakage Current (Pins 24, 27–38)			100	μA	$V_{IN} = V_{CC}$
C_{IN}	Input Capacitance			10	pF	
$C_{I/O}$	I/O Capacitance			20	pF	

3

A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = V_{DD} = +5V \pm 10\%$, $V_{SS} = 0V$

DBB READ

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}\downarrow$	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}\uparrow$	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	160		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		130	ns	$C_L = 100$ pF
t_{RD}	$\overline{RD}\downarrow$ to Data Out Delay		130	ns	$C_L = 100$ pF
t_{DF}	$\overline{RD}\uparrow$ to Data Float Delay		85	ns	
t_{CY}	Cycle Time	1.25	15	μs	1–12 MHz Crystal

DBB WRITE

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}\downarrow$	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}\uparrow$	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	160		ns	
t_{DW}	Data Setup to $\overline{WR}\uparrow$	130		ns	
t_{WD}	Data Hold to $\overline{WR}\uparrow$	0		ns	

DMA AND INTERRUPT TIMING

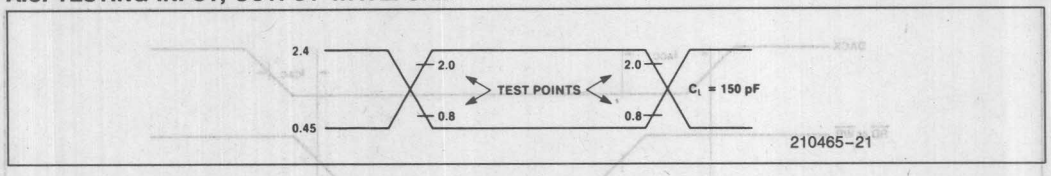
Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{ACC}	$\overline{DACK}$ Setup to Control	0		ns	
t_{CAC}	$\overline{DACK}$ Hold After Control	0		ns	
t_{ACD}	$\overline{DACK}$ to Data Valid		130	ns	$C_L = 100$ pF
t_{CRQ}	Control L.E. to DRQ T.E.		110	ns	
t_{CI}	Control T.E. to Interrupt T.E.		400	ns	

CLOCK

Symbol	Parameter	Min	Max	Units
t_{CY}	Cycle Time	1.25	9.20	μs(1)
t_{CYC}	Clock Period	83.3	613	ns
t_{PWH}	Clock High Time	38		ns
t_{PWL}	Clock Low Time	38		ns
t_R	Clock Rise Time		10	ns
t_F	Clock Fall Time		10	ns

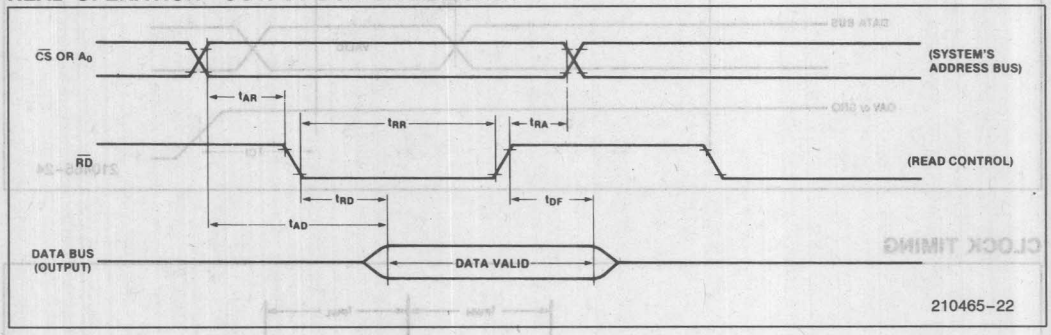
NOTE:
1. $t_{CY} = 15/f(XTAL)$

A.C. TESTING INPUT, OUTPUT WAVEFORM

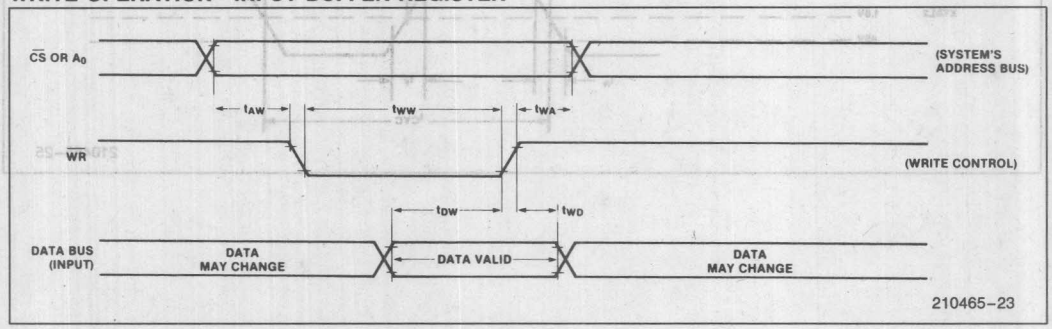


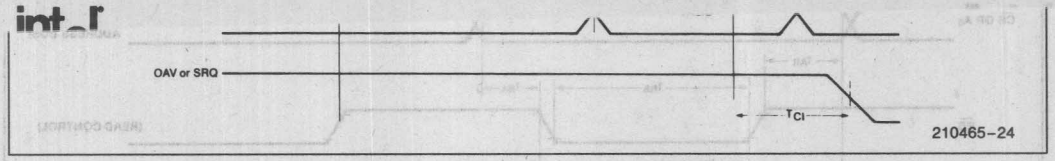
WAVEFORMS

READ OPERATION—OUTPUT BUFFER REGISTER



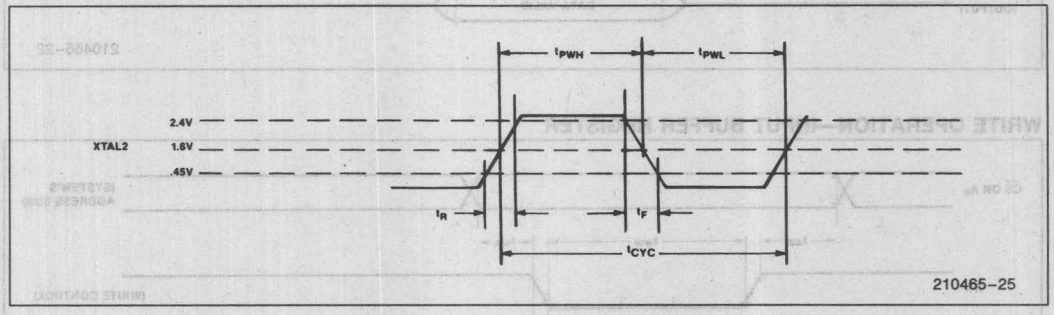
WRITE OPERATION—INPUT BUFFER REGISTER





210465-24

CLOCK TIMING



210465-25

October 1990

Using the 8291A GPIB Talker/Listener

Order Number: 230832-001

USING THE 8291A GPIB TALKER/LISTENER

CONTENTS

PAGE

INTRODUCTION 3-67

OVERVIEW OF IEEE 488/GPIB 3-68
Interface Functions 3-68
Electrical Signal Lines 3-68
Transfer Control Lines 3-70
Bus Commands 3-70
Addressing Techniques 3-70

INTEL'S ® GPIB COMPONENTS 3-70
Overview of the 8291A GPIB Talker/Listener 3-71
Address Mode 3-72

APPLICATION OF THE 8291A 3-73
Talker Functions 3-73
Addressed Talker (via MTA Message) ... 3-73

LISTENER FUNCTIONS 3-73
Addressed Listening (via the MLA Message) 3-73
Remote/Local and Lockout 3-73
Polling 3-74
Serial Poll 3-74
Parallel Poll 3-76

APPLICATION EXAMPLES 3-77
Two Software Drivers 3-77
8291A with HP 9835A 3-77

CONCLUSION 3-78

REFERENCES 3-78

APPENDIX A: SYSTEM BLOCK DIAGRAM WITH 8088 3-79

APPENDIX B: SOFTWARE DRIVERS FOR BLOCK DATA TRANSFER 3-80

APPENDIX C: SOFTWARE FOR HP 9835A 3-88

APPENDIX D: SOFTWARE FOR HP 8088/HP 9835A VIA GPIB 3-89

INTRODUCTION

This application note explains the Intel 8291A GPIB (General Purpose Interface Bus) Talker/Listener as a component, and shows its use in GPIB interface design tasks.

The first section of this note presents an overview of IEEE 488 (GPIB). The second section introduces the Intel GPIB component family. A detailed explanation of the 8291A follows. Finally, some application examples using the component family are presented.

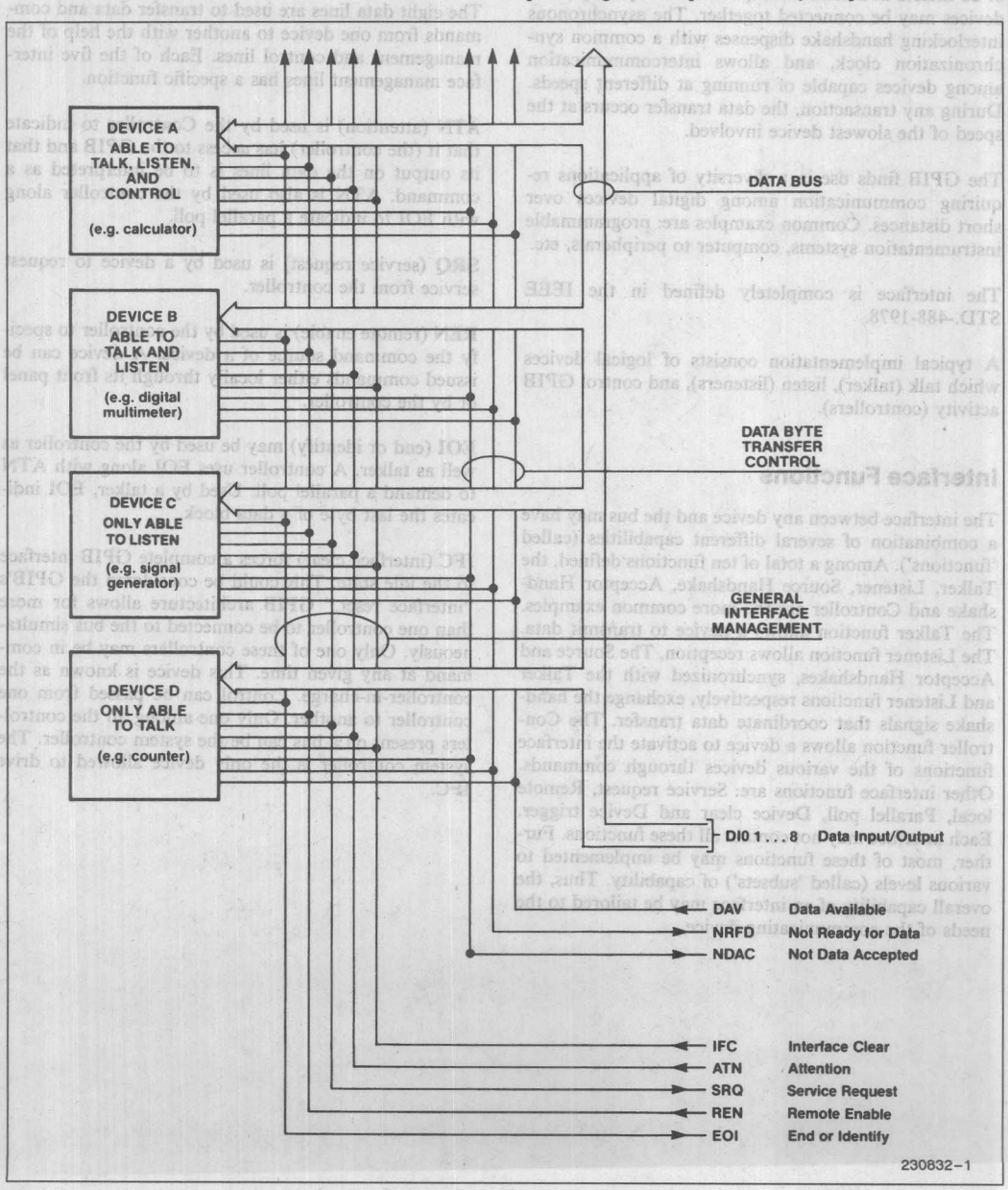


Figure 1. Interface Capabilities and Bus Structure

short distances. Common examples are: programmable instrumentation systems, computer to peripherals, etc.

The interface is completely defined in the IEEE STD.-488-1978.

A typical implementation consists of logical devices which talk (talker), listen (listeners), and control GPIB activity (controllers).

Interface Functions

The interface between any device and the bus may have a combination of several different capabilities (called 'functions'). Among a total of ten functions defined, the Talker, Listener, Source Handshake, Acceptor Handshake and Controller are the more common examples. The Talker function allows a device to transmit data. The Listener function allows reception. The Source and Acceptor Handshakes, synchronized with the Talker and Listener functions respectively, exchange the handshake signals that coordinate data transfer. The Controller function allows a device to activate the interface functions of the various devices through commands. Other interface functions are: Service request, Remote local, Parallel poll, Device clear and Device trigger. Each interface may not contain all these functions. Further, most of these functions may be implemented to various levels (called 'subsets') of capability. Thus, the overall capability of an interface may be tailored to the needs of the communicating device.

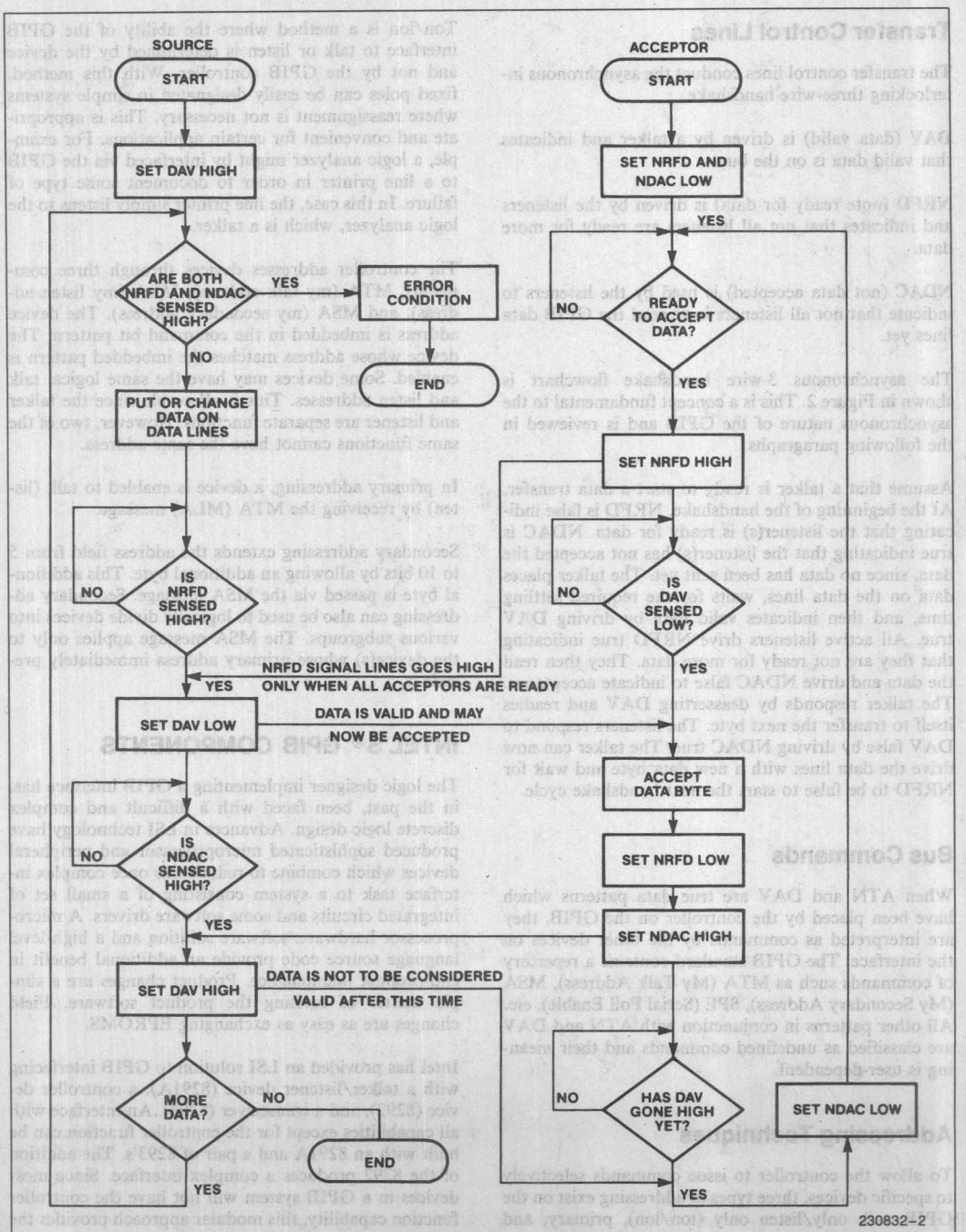
with EOI to indicate a parallel poll.

SRQ (service request) is used by a device to request service from the controller.

REN (remote enable) is used by the controller to specify the command source of a device. A device can be issued commands either locally through its front panel or by the controller.

EOI (end or identify) may be used by the controller as well as talker. A controller uses EOI along with ATN to demand a parallel poll. Used by a talker, EOI indicates the last byte of a data block.

IFC (interface clear) forces a complete GPIB interface to the idle state. This could be considered the GPIB's "interface reset." GPIB architecture allows for more than one controller to be connected to the bus simultaneously. Only one of these controllers may be in command at any given time. This device is known as the controller-in-charge. Control can be passed from one controller to another. Only one among all the controllers present on a bus can be the system controller. The system controller is the only device allowed to drive IFC.



NOTE:
Flow diagram outlines sequence of events during transfer of data byte. More than one listener at a time can accept data because of logical connection of NRFD and NDAC lines.

Figure 2. Handshake Flowchart

Transfer Control Lines

The transfer control lines conduct the asynchronous interlocking three-wire handshake.

DAV (data valid) is driven by a talker and indicates that valid data is on the bus.

NRFD (note ready for data) is driven by the listeners and indicates that not all listeners are ready for more data.

NDAC (not data accepted) is used by the listeners to indicate that not all listeners have read the GPIB data lines yet.

The asynchronous 3-wire handshake flowchart is shown in Figure 2. This is a concept fundamental to the asynchronous nature of the GPIB and is reviewed in the following paragraphs.

Assume that a talker is ready to start a data transfer. At the beginning of the handshake, NRFD is false indicating that the listener(s) is ready for data. NDAC is true indicating that the listener(s) has not accepted the data, since no data has been sent yet. The talker places data on the data lines, waits for the required settling time, and then indicates valid data by driving DAV true. All active listeners drive NRFD true indicating that they are not ready for more data. They then read the data and drive NDAC false to indicate acceptance. The talker responds by deasserting DAV and readies itself to transfer the next byte. The listeners respond to DAV false by driving NDAC true. The talker can now drive the data lines with a new data byte and wait for NRFD to be false to start the next handshake cycle.

Bus Commands

When ATN and DAV are true data patterns which have been placed by the controller on the GPIB, they are interpreted as commands by the other devices on the interface. The GPIB standard contains a repertory of commands such as MTA (My Talk Address), MSA (My Secondary Address), SPE (Serial Poll Enable), etc. All other patterns in conjunction with ATN and DAV are classified as undefined commands and their meaning is user-dependent.

Addressing Techniques

To allow the controller to issue commands selectively to specific devices, three types of addressing exist on the GPIB: talk only/listen only (ton/lon), primary, and secondary.

Ton/lon is a method where the ability of the GPIB interface to talk or listen is determined by the device and not by the GPIB controller. With this method, fixed poles can be easily designated in simple systems where reassignment is not necessary. This is appropriate and convenient for certain applications. For example, a logic analyzer might by interfaced via the GPIB to a line printer in order to document some type of failure. In this case, the line printer simply listens to the logic analyzer, which is a talker.

The controller addresses devices through three commands, MTA (my talk address), MLA (my listen address), and MSA (my secondary address). The device address is imbedded in the command bit pattern. The device whose address matches the imbedded pattern is enabled. Some devices may have the same logical talk and listen addresses. This is allowable since the talker and listener are separate functions. However, two of the same functions cannot have the same address.

In primary addressing, a device is enabled to talk (listen) by receiving the MTA (MLA) message.

Secondary addressing extends the address field from 5 to 10 bits by allowing an additional byte. This additional byte is passed via the MSA message. Secondary addressing can also be used to logically divide devices into various subgroups. The MSA message applies only to the device(s) whose primary address immediately precede it.

INTEL'S® GPIB COMPONENTS

The logic designer implementing a GPIB interface has, in the past, been faced with a difficult and complex discrete logic design. Advances in LSI technology have produced sophisticated microprocessor and peripheral devices which combine to reduce this once complex interface task to a system consisting of a small set of integrated circuits and some software drivers. A microprocessor hardware/software solution and a high-level language source code provide an additional benefit in end-product maintenance. Product changes are a simple matter of revising the product software. Field changes are as easy as exchanging EPROMS.

Intel has provided an LSI solution to GPIB interfacing with a talker/listener device (8291A), a controller device (8292), and a transceiver (8293). An interface with all capabilities except for the controller function can be built with an 8291A and a pair of 8293's. The addition of the 8292 produces a complex interface. Since most devices in a GPIB system will not have the controller function capability, this modular approach provides the least cost to the majority of interface designs.

Overview of the 8291A GPIB Talker/Listener

The Intel 8291A GPIB Talker/Listener operates over a clock range of 1 to 8 MHz and is compatible with the MCS-85, iAPX-86, and 8051 families of microprocessors.

A detailed description of the 8291A is given in the data sheet.

The 8291A implements the following functions: Source Handshake (SH), Acceptor Handshake (AH), Talker Extended (TE), Service Request (SRQ), Listener Extended (LE), Remote/Local (RL), Parallel Poll (PP2), Device Clear (DC), and Device Trigger (DT).

Current states of the 8291A can be determined by examining the device's status read registers. In addition, the 8291A contains 8 write registers. These registers are shown in Figure 3. The three register select pins RS3–RS0 are used to select the desired register.

The data-in register moves data from the GPIB to the microprocessor or to memory when the 8291A is addressed to listen. When the 8291A is addressed to talk, it uses the data-out register to move data onto the GPIB. The serial poll mode and status registers are used to request service and program the serial poll status byte.

A detailed description of each of the registers, along with state diagrams can be found in the 8291A data sheet.

Read Registers								Register Select Code			Write Registers							
								RS2	RS1	RS0								
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	0	0	0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
DATA IN											DATA OUT							
CPT	APT	GET	END	DEC	ERR	BO	BI	0	0	1	CPT	APT	GET	END	DEC	ERR	BO	BI
INTERRUPT STATUS 1											INTERRUPT ENABLE 1							
INT	SPAS	LLO	REM	SPC	LLOC	REMC	ADSC	0	1	0	0	0	DMAO	DMAI	SPC	LLOC	REMC	ADSC
INTERRUPT STATUS 2											INTERRUPT ENABLE 2							
S8	SRQS	S6	S5	S4	S3	S2	S1	0	1	1	S8	RSV	S6	S5	S4	S3	S2	S1
SERIAL POLL STATUS 2											SERIAL POLL MODE							
ton	Lon	EOI	LPAS	TPAS	LA	TA	MJMN	1	0	0	TO	LO	0	0	0	0	ADM1	ADM0
ADDRESS STATUS											ADDRESS MODE							
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	1	0	1	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
COMMAND PASS THROUGH											AUX MODE							
INT	DTO	DLO	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	1	1	0	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
ADDRESS 0											ADDRESS 0/1							
X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	1	1	1	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
ADDRESS 1											EOS							

Figure 3. 8291A Registers

Address Mode

The address mode and status registers are used to program the addressing modes and track addressing states. The auxiliary mode register is used to select a variety of functions. The command pass through register is used for undefined commands and extended addresses. The address 0/1 register is used to program the addresses to which the 8291A will respond. The address 0 and address 1 registers allow reading of these programmed addresses plus trading of the interrupt bit. The EOS register is used to program the end of sequence character.

Detailed descriptions of the addressing modes available with the 8291A are described in the 8291A data sheet. Examples of how to program these modes are shown below.

1. MODE: Talker has single address of 01H
Listener has single address of 02H

CPU Writes to:	Pattern	Comment
Address Mode Register	0000 0001	Select Mode 1 Addressing
Address 0/1 Register	0010 0001	Major is Talking. Address = 01H
Address 0/1 Register	1100 0010	Minor is Listener. Address = 02H

2. MODE: Talker has single address of 01H
Listener has single address of 02H

CPU Writes to:	Pattern	Comment
Address Mode Register	0000 0001	Select Mode 1 Addressing
Address 0/1 Register	0100 0010	Major is Listener. Address = 02H
Address 0/1 Register	1010 0001	Minor is Talking. Address = 01H

Note that in both of the above examples, the listener will respond to a MLA message with five least significant bits equal to 02H and the talker to a 01H.

3. MODE: Talker and listener both share a single address of 03H

CPU Writes to:	Pattern	Comment
Address Mode Register	0000 0001	Select Mode 1 Addressing
Address 0/1 Register	0000 0011	Talker and Listener Address = 03
Address 0/1 Register	1110 0000	Minor Address is disabled

4. MODE: Talker and listener have a primary address of 04H and a secondary address of 05H

CPU Writes to:	Pattern	Comment
Address Mode Register	0000 0010	Select Mode 2 Addressing
Address 0/1 Register	0000 0100	Primary Address = 04H
Address 0/1 Register	1000 0101	Minor Address is disabled

5. MODE: Talker has a primary address of 06H. Listener has a primary address of 07H

CPU Writes to:	Pattern	Comment
Address Mode Register	0000 0011	Select Mode 3
Address 0/1 Register	0010 0110	Talker Address = 06
Address 0/1 Register	1100 0111	Listener Primary = 07

The CPU will verify the secondary addresses which could be the same or different.

APPLICATION OF THE 8291A

This phase of the application note will examine programming of the 8291A, corresponding bus commands and responses, CPU interruption, etc. for a variety of GPIB activities. This should provide the reader with a clear understanding of the role of the 8291A performs in a GPIB system. The talker function, listener function, remote message handling, and remote/local operations including local lockout, are discussed.

Talker Functions

TALK-ONLY (ton). In talk only mode the 8291A will not respond to the MTA message from a controller. Generally, ton is used in an environment which does not have a controller. Ton is also employed in an interface that includes the controller function.

When the 8291A is used with the 8292, the sequence of events for initialization are as follows:

1) The Interrupt/Enable registers are programmed.
2) Ton is selected.
3) Settling time is selected.
4) EOS character is loaded.
5) "Pon" local message is sent.
6) CPU waits for Byte Out (BO) and sends a byte to the data out register.

Addressed Talker (via MTA Message)

The GPIB controller will direct the 8291A to talk by sending a My Talk Address (MTA) message containing the 8291A's talk address. The sequence of events is as follows:

1) The interrupt enable and serial poll mode registers are programmed.
2) Mode 1 is selected.
3) Settling time is selected.
4) Talker and listener addresses are programmed.
5) Power on (pon) local message is sent.
6) CPU waits for an interrupt. When the controller has sent the MTA message for the 8291A an interrupt will be generated if enabled and the ADSC bit will be set.
7) CPU reads the Address Status register to determine if the 8291A has been addressed to talk (TA = 1).
8) CPU waits for an interrupt from either BO or ADSC
9) When BO is set, the CPU writes the data byte to the data out register.
10) CPU continues to poll the status registers.
11) When unaddressed ADSC, will be set and TA reset.

LISTENER FUNCTIONS

LISTEN-ONLY (lon). In listen-only mode the 8291 will not respond to the My Listen Address (MLA) message from the controller. The sequence of events is as follows:

1) The Interrupt Enable registers are programmed.
2) Lon is selected.
3) EOS character is programmed.
4) "Pon" local message is sent.
5) CPU waits for BI and reads the byte from the data-in register.

Note that enabling both ton and lon can create an internal loopback as long as another listener exists.

Addressed Listening (via the MLA Message)

The GPIB controller will direct the 8291A to listen by sending a MLA message containing the 8291A's listen address. The sequence of events is as follows:

1) The Interrupt Enable registers are programmed.
2) The serial poll mode register is loaded as desired.
3) Talker and listener addresses are loaded.
4) "Pon" local message is sent.
5) The CPU waits for an interrupt. When the controller has sent the MLA message for the 8291A, the ADSC bit will be set.
6) The CPU reads the Address Status Register to determine if the 8291A has been addressed to listen (LA = 1).
7) CPU waits for an interrupt for BI or ADSC.
8) When BI is set, the CPU reads the data byte from the data-in register.
9) The CPU continues to poll the status registers.
10) When unaddressed, ADSC will be set and LA reset.

Remote/Local and Lockout

Remote and local refer to the source of control of a device connected to the GPIB. Remote refers to control from the GPIB controller-in-charge. Local refers to control from the device's own system. Reference should be made to the RL state diagram in the 2891A data sheet.

Upon "pon" the 8291A is in the local state. In this state the REM bit in Interrupt Status 1 Register is reset. When the GPIB controller takes control of the bus it will drive the REN (remote enable) line true. This will cause the REM bit and REMC (remote/local change) bit to be set. The distinction between remote and local modes is necessary in that some types of devices will have local controls which have functions which are also controlled by remote messages.

In the local state the device is allowed to store, but not respond to, remote messages which control functions which are also controlled by local messages. A device which has been addressed to listen will exit the local state and go to the remote state if the REN message is true and the local rtl (return to local) message is false. The state of the "rtl" local message is ignored and the device is "locked" into the local state if the LLO remote message is true. In the Remote state the device is not allowed to respond to local messages which control function that are also controlled by remote messages. A device will exit the remote state and enter the local state when REN goes false. It will also enter the local state if the GTL (go to local) remote message is true and the device has been addressed to listen. It will also enter the local state if the rtl message is true and the LLO message is false or ACDS is inactive.

A device will exit the remote state and enter RWLS (remote with lockout state) if the LLO (local lockout) message is true and ACDS is active. In this mode, those local messages which control functions which are also controlled by remote messages are ignored. In other words, the "rtl" message is ignored. A device will exit RWLS and go to the local state if REN goes false. The device will exit RWLS and go to LWLS if the GTL message is true and the device is addressed to listen.

Polling

The IEEE-488 standard specifies two methods for a slave device to let the controller know that it needs service.

These two methods are called Serial and Parallel Poll. The controller performs one of these two polling methods after a slave device requests service. As implied in the name, a Serial Poll is when the controller sequentially asks each device if it requested service. In a Parallel Poll the controller asks all of the devices on the GPIB if they requested service, and they reply in parallel.

Serial Poll

When the controller performs a Serial Poll, each slave device sends back to the controller a Serial Poll Status Byte. One of the bits in the Serial Poll Status Byte indicates whether this device requested service or not. The remaining 7 bits are used defined, and they are used to indicate what type of service is required. The IEEE-488 spec only defines the service request bit, however HP has defined a few more bits in the Serial Poll Status Byte. This can be seen in Figure 4.

When a slave device needs service it drives the SRQ line on the GPIB bus true (low). For the 8291A this is done by setting bit 7 in the Serial Poll Status Byte. The CPU in the controller may be interrupted by SRQ or it may poll a register to determine the state of SRQ. Using the 8292 one could either poll the interrupt status register for the SRQ interrupt status bit, or enables SRQ to interrupt the CPU. After the controller recognizes a service request, it goes into the serial poll routine.

The first thing the controller does in the serial poll routine is assert ATN. When ATN is asserted true the controller takes control of the GPIB, and all slave de-

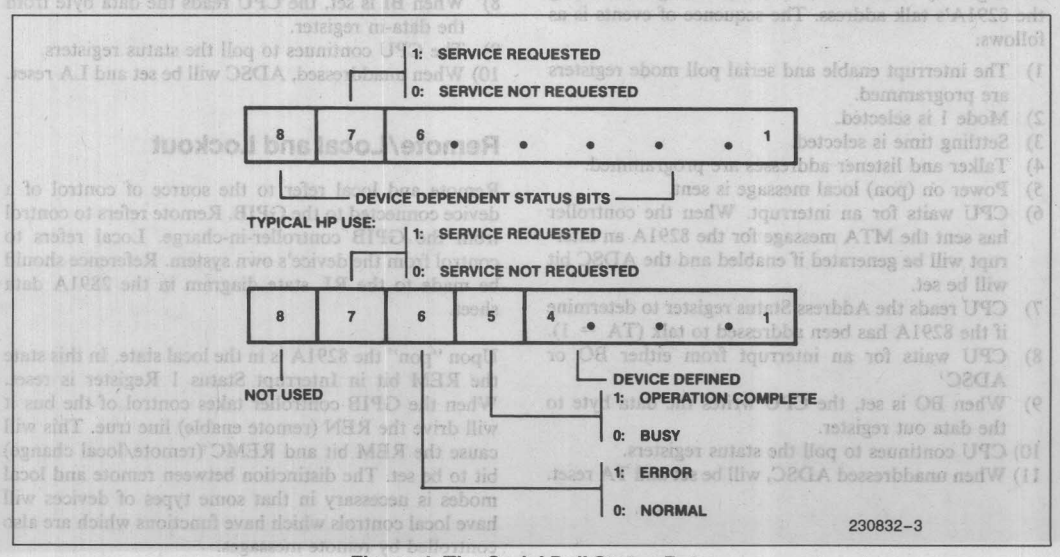


Figure 4. The Serial Poll Status Byte

vices on the bus must listen. All bytes sent over the bus while ATN is true are commands. After the controller takes control, it sends out a Universal Unlisten (UNL), which tells all previously addressed listeners to stop listening. The controller then sends out a byte called SPE (Serial Poll Enable). This command notifies all of the slaves on the bus that the controller has put the GPIB in the Serial Poll Mode State (SPMS). Now the controller addresses the first slave device to TALK and puts itself in the listen mode. When the controller resets ATN the device addressed to talk transmits to the controller its Serial Poll Status Byte. If the device just polled was the one requesting service, the SRQ line on the GPIB goes false, and bit 7 in the serial poll status byte of the 8291A is reset. If more than one device is requesting service, SRQ remains low until all of the devices requesting service have been polled, since SRQ is wire-ored. To continue the Serial Poll, the controller asserts ATN, addresses the next device to talk then reads the Serial Poll Status Byte. When the controller is finished polling it asserts ATN, sends the univeral untalk command (UNT), then sends the Serial Poll Disable command (SPD). The flow of the serial poll can be seen from the example in Figure 5.

0) DEVICE A REQUESTS SERVICE (SRQ)
1) ASSERT ATN
2) UNIVERSAL UNLISTEN (UNL)
3) SERIAL POLL ENABLE (SPE)
4) DEVICE A TALK ADDRESS (MTA)
5) RELEASE ATN
6) DEVICE A STATUS BYTE (STD) (RQS SET)
7) ASSERT ATN
8) DEVICE B TALK ADDRESS (MTA)
9) RELEASE ATN
10) DEVICE B STATUS BYTE (STB) (RQS CLEAR)
11) ASSERT ATN
12) DEVICE C TALK ADDRESS (MTA)
13) RELEASE ATN
14) DEVICE C STATUS BYTE (STB) (RQS CLEAR)
15) ASSERT ATN
16) UNIVERSAL UNTALK (UNT)
17) SERIAL POLL DISABLE (SPD)
18) GO PROCESS SERVICE REQUEST

Figure 5. Serial Polling

The following section describes the events which happen in a serial poll when 8291A and 8292 are the controller, and another 8291A is the slave device. While going through this section the reader should refer to the register diagrams for the 8291A and 8292.

A. DEVICE A REQUESTS SERVICE (SRQ BECOMES TRUE)

The slave devices rsv bit in the 2819A's serial poll mode register is set.

B. CONTROLLER RECOGNIZES SRQ AND ASSERTS ATN

The 8292's SPI pin 33 interrupts the CPU. The CPU reads the 8292's Interrupt status register and finds the SRQ bit set. The CPU tells the 8292 to 'Take Control Synchronously' by writing a OFDH to the 8292's command register.

C. THE CONTROLLER SENDS OUT THE FOLLOWING COMMANDS: UNIVERSAL UNLISTEN (UNL), SERIAL POLL ENABLE (SPE), MY TALK ADDRESS (MTA)

(MTA is a command which tells one of the devices on the bus to talk.)

The CPU in the controller waits for a BO (byte out) interrupts in the 8291A's interrupt status 1 register before it writes to the Data Out register a 3FH (UNL), 18H (SPE), 010XXXXX (MTA). The X represents the programmable address of a device on the GPIB. When the 8291A in the slave device receives its talk address, the ADSC bit in the Interrupt Status register 2 is set, and in the Address Status Register TA and TPAS bits are set.

D. CONTROLLER RECONFIGURES ITSELF TO LISTEN AND RESETS ATN

The CPU in the controller puts the 8291A in the listen only mode by writing a 40H to the Address Mode register of the 8291A, and then a OOH to the Aux Mode register. The second write is an 'Immediate Execute pon' which must be used when switching addressing modes such as talk only to listen only. To reset ATN the CPU tells the 8292 to 'Go To Standby' by writing a 0F6H to the command register. The moment ATN is reset, the 8219A in the slave device sets SPAS in Interrupt Status 2 register, and transmits the serial poll status byte. SRQS in the Serial Poll Status byte of the 8291A slave device is reset, and the SRQ line on the GPIB bus becomes false.

E. THE CONTROLLER READS THE SERIAL POLL STATUS BYTE, SETS ATN, THEN RECONFIGURES ITSELF TO TALK

The CPU in the controller waits for the Byte In bit (BI) in the 8291A's Interrupt Status 1 register. When this bit is set the CPU reads the Data In register to receive the Serial Poll Status Byte. Since bit 7 is set, this was the device which requested service. The CPU in the controller tells the 8292 to 'Take Control Synchronously' which asserts ATN. The moment ATN is asserted true the 8291A in the slave device resets SPAS, and sets the

Serial Poll Complete (SPC) bit in the Interrupt Status 2 register. The controller reconfigures itself to talk by setting the TO bit in the Address Mode register and then writing a OOH to the Aux Mode register.

F. THE CONTROLLER SENDS THE COMMANDS UNIVERSAL UNTALK (UNT), AND SERIAL POLL DISABLE (SPD) THEN RESETS THE SRQ BIT IN THE 8292 INTERRUPT STATUS REGISTER

The CPU in the controller waits for the BO Interrupt status bit to be set in the Interrupt Status 1 register of the 8291A before it writes 5FH (UNT) and 19H (SPD) to the Data Out register. The CPU then writes a 2BH to the 8292's command register to reset the SRQ status bit in the Interrupt Status register. When the 8291A in the slave device receives the UNT command the ADSC bit in the Interrupt Status 2 register is set, and the TA and TPAS bits in the Address Status register will be reset. At this point the controller can service the slave device's request.

Note that in the software listing of AP-66 (USING THE 8292 GPIB CONTROLLER) there is a bug in the serial poll routines. In the 'SRQ ROUTINE' when the CPU finds that the SRQ bit in the interrupt status register is set, it immediately writes the interrupt Acknowledge command to the 8292 to reset this bit. However the SRQ GPIB line will still be driven true until the slave device driving SRQ has been polled. Therefore, the SRQ status bit in the 8292 will become set and latched again, and as a result the SRQ status bit in the 8292 will still be set after the serial poll. The proper time to reset the SRQ bit in the 8292 is after SRQ on the GPIB becomes false.

Parallel Poll

The 8291A supports an additional method for obtaining status from devices known as parallel poll (PPOL). This method limits the controller to a maximum of 8 devices at a time since each device will produce a single bit response on the GPIB data lines. As shown in the state diagrams, there are three basic parallel poll states: PPIS (parallel poll idle state), PPSS (parallel poll standby state), and PPAS (parallel poll active state).

In PPIS, the device's parallel poll function is in the idle state and will not respond to a parallel poll. PPSS is the standby state, a state in which the device will respond to a parallel poll from the controller. The response is initiated by the controller driving both ATN and EOI true simultaneously.

The 8291A state diagram shows a transition from PPIS to PPSS with the "lpe" message. This is a PP2 implementation for a parallel poll. This "lpe" (local poll enable) local message is achieved by writing $011USP_3P_2P_1$ to the Aux Mode Register with $U=0$. The S bit is the sense bit. If the "ist" (individual status) local message value matches the sense bit, then the 8291A will give a true response to a parallel poll. Bits P_3–P_1 identify which data line is used for a response.

For example, assume the programmer decides that the system containing the 8291A shall participate in parallel poll. The programmer, upon system initialization would write to the Aux Mode Register and reset the U bit and set the S bit plus identify a data line (P_3–P_1 bits). At "pon," the 8291A would not respond true to a parallel poll unless the parallel poll flag is set (via Aux Mode Register command).

When a status condition in the user system occurs and the programmer decides that this condition warrants a true response, then programmers software should set the parallel poll flag. Since the S bit value matches the "ist" (set) condition a true response will be given to all parallel polls.

An additional method of parallel polling reading exists known as a PP1 implementation. In this case the controller sends a PPE (parallel poll enable) message. PPE contains a bit pattern similar to the bit pattern used to program the "lpe" local message. The 8291A will receive this as an undefined command and use it to generate an "lpe" message. Thus the controller is specifying the sense bits and data lies for a response. A PPD (parallel poll disable) message exists which clears the bits $SP_3P_2P_1$ and sets the U bit. This also will be received by the 8291A and used to generate an "lpe" false local message.

The actual sequence of events is as follows. The controller sends a PPC (parallel poll configure) message. This is an undefined command which is received in the CPT register and the handshake is held off. The local CPU reads this bit pattern, decodes it, and sends a VSCMD message to the Aux Mode Register. The controller then sends a ppe message which is also received as an undefined command in the CPT register. The local CPU reads this, decodes it clears the MSB, and writes this to the Aux Mode Register generating the "lpe" message.

The controller then sends ATN and EOI true and the 8291A drives the appropriate data line if the "ist" (parallel poll flag) is true. The controller will then send a PPD (parallel poll disable) message (again, an undefined command). The CPU reads this from the CPT register and uses it to write new "lpe" message (this "lpe" message will be false). The controller then sends a PPU (parallel poll unconfigure) message. Since this is also an undefined command, it goes into the CPT register. When the local CPU decodes this, the CPU should clear the "ist" (parallel poll flag).

APPLICATION EXAMPLES

In the course of developing this application note, two complete and identical; GPIB systems were built. The schematics and block diagrams are contained in Appendix 1. These systems feature an 8088 CPU, 8237 DMA controller, serial I/O (8215a and 8253), RAM, EPROM, and a complete GPIB talker/listener controller. Jumper switches were provided to select between a controller function and a talker/listener function. This system design is based on the design of Intel's SDK-86 prototyping kit and thus shares the same I/O and memory addresses. This system uses the same download software to transfer object files from Intel development systems.

Two Software Drivers

Two software drivers were developed to demonstrate a ton/lon environment. These two programs (BOARD 1 and BOARD 2) are contained in Appendix 2.

In this example, one of the systems (BOARD 1) initially is programmed in talk-only mode and synchronization is achieved by waiting for the listening board to become active. This is sensed by the lack of a GPIB error since a condition of no active listener produces an ERR status condition. Board 1 upon detecting the presence of an active listener transmits a block of 100 bytes from a PROM memory across the bus. The second system (BOARD 2) receives this data and stores it in a buffer, EOI is sent true by the talker (BOARD 1) with the last byte of data. Upon detection of EOI, BOARD 2 switches to the talk only mode while BOARD 1 upon terminal count switches to the listen only mode. BOARD 2 then detects the presence of an active listener and transmits the contents of its buffer back to BOARD 1 which stores this data in the buffer. EOI again is sent with the last byte and BOARD 2 switches back to listen-only. BOARD 1 upon detecting EOI then compares the contents of its buffer with the contents of its PROM to ensure that no data transmission errors occurred. The process then repeats itself.

8291A with HP 9835A

An example of the 8291A used in conjunction with a bus controller is also included in this application note. In this example, the 8291A system used in previous experiments was connected via the GPIB to a Hewlett-Packard 9835A desktop computer. This computer contains, in addition to a GPIB interface, a black and white CRT, keyboard, tape drive for high quality data cassettes, and a calculator type printer. The software for the HP9835S is shown in Appendix 3. The user should refer to the operation manuals for the HP 9835A for information on the features and programming methods for the HP 9835A.

In this example, the 8292 was removed from its socket and the OPTA and OPTB pins of the two 8293 transceiver reconfigured to modes 0 and 1. Optionally, the mode pins could have been left wired for modes 2 and 3 and the 8292 left in its socket with its SYC pin wired to ground. This would have produced the same effect.

The first action performed is sending IFC. Generally, this is done when a controller first comes on line. This pulse is at least 100 μs in duration as specified by the IEEE-488 standard.

The software checks to see if active listeners are on line. For demonstration purposes, the HP 9835A will flag the operator to indicate that listeners are on line.

The HP 9835A then configures and performs a parallel poll (PPOL). The parallel poll indicates 1 bit of status of each device in a group of up to 8 devices. Such information could be used by an application program to determine whether optional devices are part of a system configuration. Such optional devices might include mass storage devices, printers, etc., where the application software for the controller might need to format data to match each type of device. Once the PPOL sequence is finished, the HP 9835A offers the user the opportunity to execute user commands from the keyboard. At this time the HP 9835A sits in a loop waiting for an SRQ condition. When the operator hits a key on the keyboard, the HP 9835A processor is interrupted and vectors to a service routine where the key is read and the appropriate routine is executed. The HP 9835A will then return to the loop checking for the SRQ true. For this application, the valid keys are G, D, R, H, and X. Pressing the "G" key causes the GET command to be sent across the bus. A message to this effect is printed in the CRT and the HP 9835A returns. The "D" key causes the SDC message to be sent with the 8291A being the addressed device. Again, an appropriate message is output on the HP 9835A CRT. The "R" key causes the GTL message to be sent. The CRT displays "REMOTE MESSAGE SENT." The "H" key causes a menu to be displayed on the HP 9835A CRT screen. This menu lists the allowed commands and their functions. NO GPIB commands are sent. The "X" key allows the operator to send one line of data across the bus. The line of data is terminated by a carriage return and line feed produced by pressing the "CONTINUE" key on the HP 9835A.

The characters are stored in the sequence entered into a buffer whose maximum size is 80 characters. Pressing the "CONTINUE" key terminates storing characters in the array and all characters including the carriage return and line feed are sent. EOI is then sent true with a false byte of OOH. This false byte is due to the 1975 standard which allows asynchronous sending and reception of EOI. (The 8291A supports the later 1978 standard which eliminates this false byte.)

After any key command is serviced control returns to the loop which checks for SRQ active. Should SRQ be active, then the keyboard interrupt is disabled and a message printed to indicate that SRQ has been received true.

The controller then performs a parallel poll.

This is an example of how parallel poll may be used to quickly check which group of devices contains a device sending SRQ. The eight devices in a group would, of course, have software drivers which allow a true response to a PPOL if that device is currently driving SRQ true. This would be a valuable method of isolation of the SRQ source in a system with a large number of devices. In this application program, only the response from the 8291A is of concern and only the 8291A's response is considered. It does, however, demonstrate the technique employed. If a true response from the 8291A is detected, then a message to this effect is printed on the HP 9835A CRT screen. From this process, the controller has identified the device requesting service and will use a serial poll (SPOL) to determine the reason for the service request. This method of using PPOL is not specifically defined by the IEEE-488 standard but is a use of the resources provided.

The controller software then prints a message to indicate that it is about to perform a serial poll. This serial poll will return to the controller the current status of the 2819A and clear the service request. The status byte received is then printed on the CRT screen of the HP 9835A. One of the 8291A status bits indicates that the 8291A system has a field (on line or less) of data to transfer to the HP 9835A. If this bit is set, then the HP 9835A addresses the 8291A system to talk. The data is sent by the 8291A system is then printed on the CRT screen of the HP 9835A. The HP 9835 then enables the keyboard interrupts and goes into its SRQ checking loop.

Appendix 4 contains the software for the 8291A system which is connected to the HP 9835A via the GPIB. This software throws away the first byte of data it receives since this transfer was used by the HP 9835A to test when the 8291A system came on line.

Next, both status registers are read and stored in the two variable STAT 1 and STAT 2. It is necessary to store the status since reading the status registers clears the status bits.

Initially, six status bits are evaluated (END, GET, CPT, DEC, REMC, ADSC). Some of these conditions require that additional status bits be evaluated.

If END is true, then the 8291A system has received a block from the HP 9835A and the contents of a buffer is printed on the CRT screen. Next, the CPT bit is checked. PPC and PPE are only valid undefined commands in this example.

Next, the GET bit is examined and if true, the CRT screen connected to the serial channel on the 8291A system prints a message to indicate that the trigger command has been received. A similar process occurs with the DEC and REMC status bits.

Address Status Chagne (ADSC) is checked to see if the 8291A has been addressed or unaddressed by the controller. If ADSC is false, then the software checks the keyboard at the CRT terminal. If ADSC is set, then the TA and LA bits are read and evaluated to determine whether the 8291A has been addressed to talk or listen. The DMA controller is set to start transfers at the start of the character buffer and the type of transfer is determined by whether the 8291A in in TADS or LADS. We only need to set up the DMA controller since the transfers will be transparent to the system processor. The keyboard from the CRT terminal is then checked. If a key has been hit, then this character is stored in the character buffer and the buffer printer set to the next character location. This process repeats until the received character is a line feed. The line feed is echoed to the CRT, the serial poll status byte updated and the SRQ line driven true. This allows the 8291A system to store up to one line of characters before requesting a transfer to the controller. Recall that upon receiving an SRQ, the controller will perform a serial poll and subsequently address the 8291A to talk. The 8291A system then goes back to reading the status register thus repeating the process.

CONCLUSION

This application note has shown a basic method to view the IEEE 488 bus, when used in conjunction with Intel's 8291A.

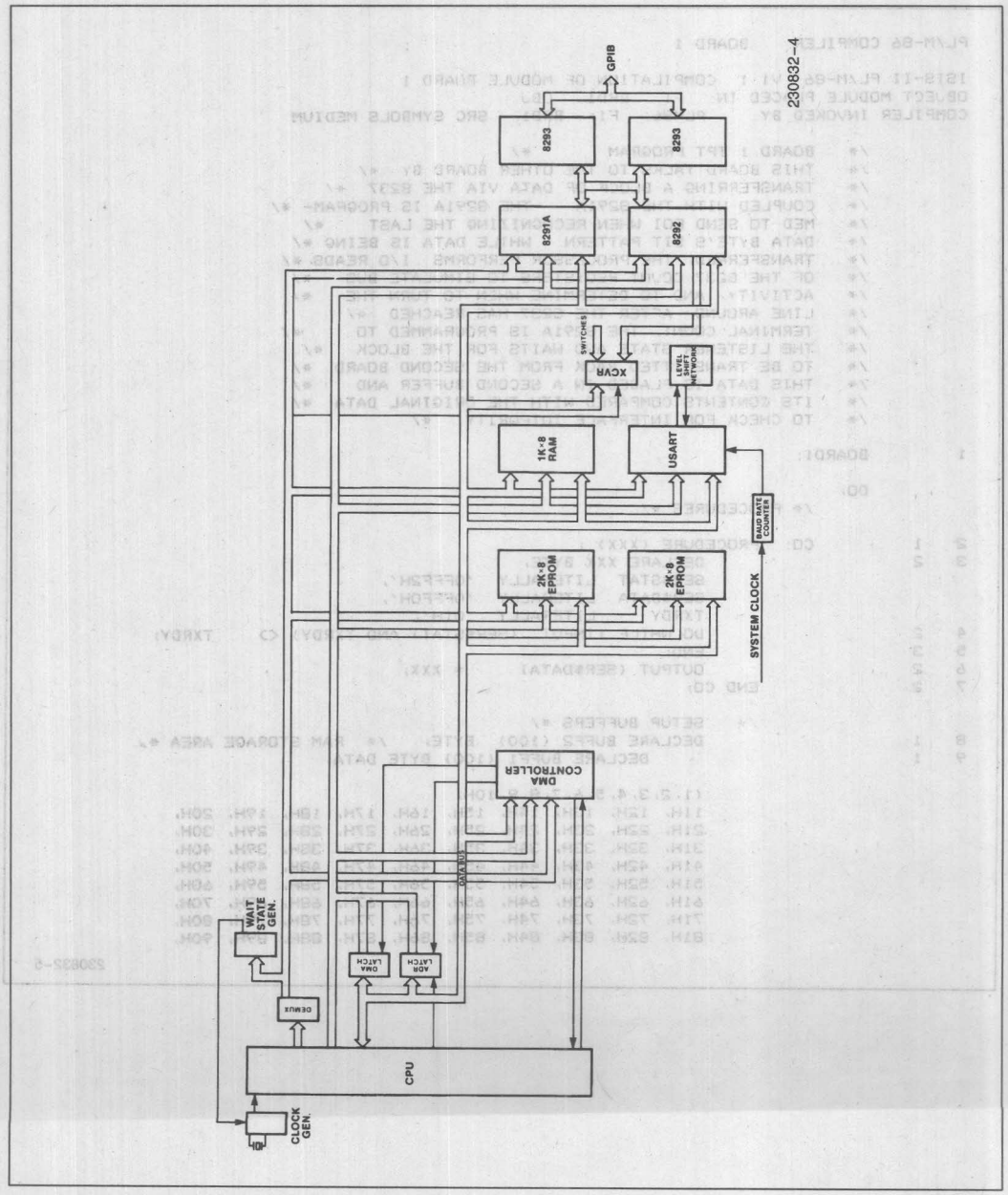
The main reference for GPIB questions is the IEEE Standard 488—1978. Reference 8291A's data sheet for detailed information on it.

Additional Intel GPIB products include iSBX-488, which is a multimode board consisting of the 8291A, 8292, and 8293.

REFERENCES

8291A Data Sheet
8292 Data Sheet
8293 Data Sheet
Application Note #66 "Using the 8292 GPIB Controller"
PLM-86 User Manual
HP 9835A User's Manual
IEEE—488—1978 Standard

APPENDIX A
SYSTEM BLOCK DIAGRAM WITH 8088



APPENDIX B
SOFTWARE DRIVERS FOR BLOCK DATA TRANSFER

```
PL/M-86 COMPILER     BOARD 1

ISIS-II PL/M-86 V1.1 COMPILATION OF MODULE BOARD 1
OBJECT MODULE PLACED IN:  F1  BRD1  OBJ
COMPILER INVOKED BY:  PLM86  F1:  BRD1.  SRC SYMBOLS MEDIUM

                /*   BOARD 1 TPT PROGRAM          */
                /*   THIS BOARD TALKS TO THE OTHER BOARD BY  */
                /*   TRANSFERRING A BLOCK OF DATA VIA THE 8237  */
                /*   COUPLED WITH THE 8291A.   THE 8291A IS PROGRAM- */
                /*   MED TO SEND EOI WHEN RECOGNIZING THE LAST    */
                /*   DATA BYTE'S BIT PATTERN.  WHILE DATA IS BEING */
                /*   TRANSFERRED, THE PROCESSOR PERFORMS  I/O READS */
                /*   OF THE 8237 COUNT REGISTERS TO SIMULATE BUS    */
                /*   ACTIVITY, AND TO DETERMINE WHEN TO TURN THE    */
                /*   LINE AROUND.  AFTER THE 8237 HAS REACHED  */
                /*   TERMINAL COUNT, THE 8291A IS PROGRAMMED TO     */
                /*   THE LISTENER STATE AND WAITS FOR THE BLOCK    */
                /*   TO BE TRANSMITTED BACK FROM THE SECOND BOARD.  */
                /*   THIS DATA IS PLACED IN A SECOND BUFFER AND     */
                /*   ITS CONTENTS COMPARED WITH THE ORIGINAL DATA  */
                /*   TO CHECK FOR INTERFACE INTEGRITY.   */

1               BOARD1:

                DO;
                     /* PROCEDURES */

2   1                CO:  PROCEDURE (XXX) ;
3   2                          DECLARE XXX BYTE,
                           SER$STAT  LITERALLY  'OFFF2H',
                           SER$DATA  LITERALLY  'OFFF0H',
                           TXRDY     LITERALLY  '01H',
4   2                      DO WHILE (INPUT  (SER$STAT) AND TXRDY) <>    TXRDY;
5   3                      END;
6   2                      OUTPUT (SER$DATA)      = XXX;
7   2                 END CO;

                      /*   SETUP BUFFERS */
8   1                 DECLARE BUFF2 (100)  BYTE;   /*  RAM STORAGE AREA */
9   1                      DECLARE BUFF1 (100) BYTE DATA

                      (1,2,3,4,5,6,7,8,9,10H,
                      11H, 12H, 13H, 14H, 15H, 16H, 17H, 18H, 19H, 20H,
                      21H, 22H, 23H, 24H, 25H, 26H, 27H, 28H, 29H, 30H,
                      31H, 32H, 33H, 34H, 35H, 36H, 37H, 38H, 39H, 40H,
                      41H, 42H, 43H, 44H, 45H, 46H, 47H, 48H, 49H, 50H,
                      51H, 52H, 53H, 54H, 55H, 56H, 57H, 58H, 59H, 60H,
                      61H, 62H, 63H, 64H, 65H, 66H, 67H, 68H, 69H, 70H,
                      71H, 72H, 73H, 74H, 75H, 76H, 77H, 78H, 79H, 80H,
                      81H, 82H, 83H, 84H, 85H, 86H, 87H, 88H, 89H, 90H,
```

230832-5

```
PL/M-86  COMPILER  BOARD1

                   91H, 92H, 93H, 94H, 95H, 96H, 97H, 98H, 99H, ODH);
10  1              DECLARE BUFF3(17)  BYTE DATA
                   (ODH, OAH, 'COMPARE ERROR', ODH, OAH);  /* ROM STORAGE AREA */

                   /* 8237 PORT ADDRESSES  */

11  1              DECLARE

                   CLEAR$FF          LITERALLY 'OFFDDH', /* MASTER CLEAR */
                   START$0$LO        LITERALLY 'OFFD0H',
                   START$0$HI        LITERALLY 'OFFD0H',
                   0$COUNT$LO        LITERALLY 'OFFD1H',
                   0$COUNT$HI        LITERALLY 'OFFD1H',
                   SET$MODE          LITERALLY 'OFFDBH',
                   CMD$37            LITERALLY 'OFFD8H',
                   SET$MASK          LITERALLY 'OFFDFH';

                   /* 8237 COMMAND  - DATA BYTES */
12  1              DECLARE           DMA$ADR$TALK    POINTER;
13  1              DECLARE           DMA$ADR$LSTN    POINTER;

14  1              DECLARE

                   RD$TRANSFER       LITERALLY '48H',
                   WR$TRANSFER       LITERALLY '44H',
                   NORM$TIME         LITERALLY '20H',
                   TC$LO1            LITERALLY 'OFFH',
                   TC$HI1            LITERALLY '00H',
                   TC$LO2            LITERALLY '99D',          /* 100 XFERS  */
                   TC                LITERALLY '01H',
                   I                 BYTE;

15  1              DECLARE

                   DMA$WRD$TALK (2)  WORD    AT    (@DMA$ADR$TALK),
                   DMA$WRD$LSTN(2)   WORD    AT    (@DMA$ADR$LSTN);
                   /* 8291A PORT ADDRESSES */

16  1              DECLARE

                   PORT$OUT          LITERALLY 'OFFC0H', /* DATA OUT*/
                   PORT$IN           LITERALLY 'OFFC0H'
                   STATUS$1          LITERALLY 'OFFC1H', /*INTR STAT 2*/
                   STATUS$2          LITERALLY 'OFFC2H', /* INTR STAT 2 */
                   ADDR$STATUS       LITERALLY 'OFFC4H',
                   COMMAND$MOD       LITERALLY 'OFFC5H', /*CMD PASS THRU */
                   ADDR$0            LITERALLY 'OFFC6H',
                   EOS$REG           LITERALLY 'OFFC7H', /*  EOS REGISTER */
```

230832-6

3

```
                /* 8291A COMMAND  - DATA BYTES */

PL/M-86 COMPILER     BOARD1

17  1               DECLARE

                    END$EOI    LITERALLY      '88H',
                    DNE   LITERALLY       '10H',
                    PON   LITERALLY       '00H',
                    RESET      LITERALLY       '02H',
                    CLEAR      LITERALLY       '00H',
                    DMA$REG$L  LITERALLY       '10H',
                    DMA$REG$T  LITERALLY       '20H',
                    MOD1$TO    LITERALLY       '80H',
                    MOD1$LO    LITERALLY       '40H',
                    EOS   LITERALLY     '0DH',
                    PRESCALER LITERALLY       '23H',
                    HIGH$SPEED LITERALLY      '0A4H',
                    OKAY       LITERALLY       'OFFFFH',
                    XYZ   BYTE,
                    MATCH      WORD,
                    BO    LITERALLY       '02H',
                    BI    LITERALLY       '01H',
                    ERR   LITERALLY       '04H';

          /* CODE BEGINS */

18  1     START91:

                    OUTPUT  (STATUS$2)  =CLEAR;  /* SHUT-OFF DMA REG BITS TO */
                                                 /* PREVENT EXTRA DMA REGS  */
                                                 /*FROM 8291A               */

                    /* MANIPULATE DMA ADDRESS VARIABLES */

19  1               DMA$ADR$TALK    =(@BUFF1);
20  1               DMA$ADR$LSTN    =(@BUFF2);
21  1               DMA$WRD$TALK(1)=SHL (DMA$WRD$TALK(1), 4);
22  1               DMA$WRD$TALK(0)=DMA$WRD$TALK (0) + DMA$WRD$TALK (1);
23  1               DMA$WRD$LSTN(1)=SHL (DMA$WRD$LSTN (1), 4);
24  1               DMA$WRD$LSTN(0)=DMA$WRD$LSTN (0) +DMA$WRD$LSTN (1);

25  1          INIT37:
                    /* INIT 8237 FOR TALKER FUNCTIONS */

26  1               OUTPUT     (CLEAR$FF)     =CLEAR; /* TOGGLE MASTER CLEAR */
27  1               OUTPUT     (CMD$37)       =NORM$TIME;
28  1               OUTPUT     (SET$MODE)     =RD$TRANSFER;
                    OUTPUT     (SET$MASK)     =CLEAR;
29  1               OUTPUT     (START$0$LO)       =DMA$WRD$TALK (0);
30  1               DMA$WRD$TALK (0)          =SHR (DMA$WRD$TALK (0), 8);
31  1               OUTPUT     (START$0$HI)       =DMA$WRD$TALK (0);
32  1               OUTPUT     (0$COUNT$LO)       =TC$LO2;
33  1               OUTPUT     (0$COUNT$HI)       =TC$HI2;
                    /* INIT 8291A FOR TALKER FUNCTIONS */

PL/M-86 COMPILER     BOARD1
```

230832-7

```
34  1              OUTPUT  (EOS$REG)     =EOS;
35  1              OUTPUT  (COMMAND$MOD)     =END$EOI; /*  EOI ON EOS SENT */
36  1              OUTPUT  (ADDR$STATUS)     =MOD1$TO; /* TALK ONLY */
37  1              OUTPUT  (COMMAND$MOD)     =PRESCALER;
38  1              OUTPUT  (COMMAND$MOD)     =HIGH$SPEED;
39  1              OUTPUT  (COMMAND$MOD)     =PON;

40  1              DO WHILE (INPUT  (STATUS$1) AND BO)  =0;
41  2              END;    /*  WAIT FOR BO INTR  */
42  1              OUTPUT  (PORT$OUT)  = 0AAH;

43  1              DO WHILE  (INPUT  (STATUS$1) AND ERR) = ERR;
44  2                 DO WHILE  (INPUT  (STATUS$1)  AND BO) = 0;
45  3                 END;  /*  WAIT FOR BO INTR  */
46  2                 OUTPUT  (PORT$OUT)  =0AAH;
47  2              END;

48  1              OUTPUT  (STATUS$2)  =DMA$REQ$T;  /*  ENABLE DMA REQS */

49  1                  DO WHILE  (INPUT  (CMD$37)  AND TC) <>         TC;
                     /* WAIT FOR TC  = 0 */
50  2                  END;

51  1              INIT37L;

                    OUTPUT  (STATUS$2)  =CLEAR;  /*  DISABLE DMA REQS */

                    /* INIT 8237 FOR LISTENER FUNCTIONS */

52  1              OUTPUT  (CLEAR$FF) 0=CLEAR; /*  TOGGLE MASTER RESET */
53  2              OUTPUT  (CMD$37)     =NORM$TIME;
54  1              OUTPUT  (SET$MODE)  =WR$TRANSFER;
55  1              OUTPUT  (SET$MASK)  =CLEAR;
56  1              OUTPUT  (START$0$LO)    =DMA$WRD$LSTN (0);
57  1              DMA$WRD$LSTN (0)     =SHR (DMA$WRD$LSTN (0), 8);
58  1
    OUTPUT      (START $0$HI)      =DMA$WRD$LSTN (0);
59  1              OUTPUT  (0$COUNT$LO)      =TC$LO1;
60  1              OUTPUT  (0$COUNT$HI)      =TC$HI1;

                    /* INIT 8291A FOR LISTENER FUNCTIONS  */

61  1              OUTPUT  (COMMAND$MOD)     =RESET;
62  1              OUTPUT  (ADDR$STATUS)     =MOD1$LO; /*  LISTEN ONLY */
63  1              OUTPUT  (COMMAND$MOD)     =PON;

64  1              DO WHILE (INPUT  (STATUS$1)  AND BI)  =0;
65  2              END;  /*  WAIT FOR BI INTR */
66  1              XYZ  = INPUT (PORT$IN);

67  1              OUTPUT  (STATUS$2)  =DMA$REQ$L;  /*  ENABLE DMA REQS  */

68  1              DO WHILE  (INPUT  (STATUS$1)  AND DNE)<>         DNE;
                    /*  WAIT FOR EOI RECEIVED  */
```

230832-8

3

```
PL/M-86 COMPILER   BOARD 1
      70   1    CMPBLK9

                /* COMPARE THE TWO BUFFERS CONTENTS */

                MATCH=CMPB  (@BUFF1, @BUFF2, 100);

      71   1    IF MATCH = OKAY THEN GOTO START91;

                /*  SEND ERROR MESSAGE IN BUFFER 3  */

      73   1    DO I=0 TO 16;
      74   2       CALL CO  (BUFF 3  (I)  );
      75   2    END;

      76   1    GOTO START91;

      77   1    END;


MODULE INFORMATION:

        CODE AREA SIZE          =01DBH    475D
        CONSTANT AREA SIZE      =0075H    117D
        VARIABLE AREA SIZE      =0070H    112D
        MAXIMUM STACK SIZE      =0006H     6D
        243 LINES READ
        0 PROGRAM ERROR (S)


END OF PL/M-86  COMPILATION
```

230832-9

```
PL/M-86 COMPILER    BOARD2

ISIS-II PL/M-86  V1.1 COMPILATION OF MODULE BOARD2
OBJECT MODULE PLACED IN    F1:  BRD2, OBJ
COMPILER INVOKED BY:     PLM86   F1:  BRD2, SRC


                    /* BOARD 2 TPT PROGRAM          */
                    /*                              */
                    /* THIS BOARD LISTENS TO THE OTHER BOARD (1)    */
                    /* AND DMA'S DATA INTO A BUFFER, WHILE WAITING */
                    /* FOR THE END INTERRUPT BIT TO BECOME ACTIVE   */
                    /* UPON END ACTIVE, THE DATA IN THE BUFFER IS   */
                    /* SENT BACK TO THE FIRST BOARD VIA THE GPID    */
                    /* WHEN THE BLOCK IS FINISHED THE 8291A IS  */
                    /* PROGRAMMED BACK INTO THE LISTENER MODE       */

    1           BOARD2:

                DO;
                    /* 8237 PORT ADDRESSES */

    2   1           DECLARE

                    CLEAR$FF        LITERALLY       'OFFDDH',     /*MASTER CLEAR */
                    START$0$Lo      LITERALLY       'OFFDOH',
                    START$0$HI      LITERALLY       'OFFDOH',
                    0$COUNT$LO      LITERALLY       'OFFD1H',
                    0$COUNT$HI      LITERALLY       'OFFD1H',
                    SET$MODE        LITERALLY       'OFFDBH',
                    CMD$37          LITERALLY       'OFFD8H',
                    SET$MASK        LITERALLY       'OFFDFH',

                    /* 8237 COMMAND  - DATA BYTES   */

    3   1           DECLARE

                    RD$TRANSFER     LITERALLY       '48H',
                    WR$TRANSFER     LITERALLY       '44H',
                    ADDR$1A         LITERALLY       'OOH',
                    ADDR$1B         LITERALLY       '01H',
                    NORM$TIME       LITERALLY       '20H',
                    TC$LO1          LITERALLY       'OFFH',
                    TC$HI1          LITERALLY       'OOH',
                    TC$LO2          LITERALLY       '99D',
                    TC$HI2          LITERALLY       'OOH',
                    TC          LITERALLY       '01H',

                    /* 8291A  PORT ADDRESSES */

    4   1           DECLARE

                    PORT$OUT        LITERALLY       'OFFCOH',
                    PORT$IN         LITERALLY       'OFFCOH',/* DATA IN */
                    STATUS$1        LITERALLY       'OFFC1H', /* INTR STAT 1 */
                    STATUS$2        LITERALLY       'OFFC2H', /* INTR STAT 2 */
                    ADDR$STATUS     LITERALLY       'OFFC4H', /* ADDR STAT   */
                    COMMAND$MOD     LITERALLY       'OFFC5H', /* CMD PASS THRU */
```

230832-10

```
                    XYZ         BYTE,
                    BO          LITERALLY '02H',
                    BI          LITERALLY '01H',
                    ERR         LITERALLY '04H';

  6   1         START91;

                    OUTPUT  (STATUS$2)  =CLEAR;  /*  END INITILIZATION STATE */

                    /* INIT 8237 FOR LISTENER FUNCTION  */

  7   1         INIT37L;

                    OUTPUT  (CLEAR$FF)  =CLEAR;  /*  TOGGLE MASTER RESET  */
  8   1             OUTPUT  (CMD$37)    =NORM$TIME;
  9   1             OUTPUT  (SET$MODE)  =WR$TRANSFER,   /*  BLOCK XFER MODE  */
 10   1             OUTPUT  (SET$MASK)  =CLEAR;
 11   1             OUTPUT  (START$0$LO)     =ADDR$1A;
 12   1             OUTPUT  (START$0$HI)     =ADDR$1B;
 13   1             OUTPUT  (0$COUNT$LO)     =TC$LO1;
 14   1             OUTPLUT (0$COUNT$HI)     =TC$HI1;

                    /*  INIT 8291A  FOR LISTENER FUNCTIONS   */

 15   1             OUTPUT  (COMMAND$MOD)    =RESET;
 16   1             OUTPUT  (ADDR$STATUS)    =MOD1$LO;
 17   1             OUTPUT  (COMMAND$MOD)    =PON;
 18   1             DO WHILE (INPUT  (STATUS$1)  AND BI)  =0;
 19   2             END;  /*  WAIT FOR BI INTR */
 20   1             XYZ= INPUT  (PORT$IN);
 21   1             OUTPUT   (STATUS$2)     =DMA$REG$L;

                    /* WAIT UNTIL EOI RCVD AND END INTR-BIT SET  */

 22   1              DO WHILE  (INPUT  (STATUS$1)  AND DNE ) <>   DNE;
```

230832-11

```
PL/M-86  COMPILER  BOARD2

23  1               END;

24  1          INIT37T:
                 /* INIT 8237 FOR TALKER FUNCTION  */

                OUTPUT  (STATUS$2)  =CLEAR;    /* CLEAR  8291A  DRQ  */
25  1           OUTPUT  (CLEAR$FF)  =CLEAR;
26  1           OUTPUT  (CMD$37)    =NORM$TIME;
27  1           OUTPUT  (SET$MODE)  =RD$TRANSFER;   /*  BLOCK XFER MODE */
28  1           OUTPUT  (SET$MASK)  =CLEAR;
29  1           OUTPUT  (START$O$LO)      =ADDR$1A;
30  1           OUTPUT  (START$O$HI)      =ADDR$1B;
31  1           OUTPUT  (O$COUNT$LO)      =TC$LO2;
32  1           OUTPUT  (O$COUNT$HI)      =TC$HI2;

                /*  INIT 8291A  FOR TALKER FUNCTION  */

33  1           OUTPUT  (EOS$REG)     =EOS;
34  1           OUTPUT  (COMMAND$MOD)         =END$EOI; /* EOI ON EOS SENT  */
35  1           OUTPUT  (ADDR$STATUS)         =MOD1$TO; /* TALK ONLY  */
36  1           OUTPUT  (COMMAND$MOD)         =PRESCALER;
37  1           OUTPUT  (COMMAND$MOD)         =HIGH$SPEED;
38  1           OUTPUT  (COMMAND$MOD)         =PON;

39  1           DO WHILE  (INPUT  (STATUS$1)   AND B0)   =0;
40  2           END;     /* WAIT FOR B0 INTR  */
41  1           OUTPUT  (PORT$OUT)   =0AAH;

42  1           DO WHILE  (INPUT  (STATUS$1)  AND ERR)   =ERR;
43  2             DO WHILE   (INPUT  (STATUS$1)   AND B0)   =0;
44  3             END;     /* WAIT FOR B0 INTR  */
45  2             OUTPUT  (PORT$OUT)  =0AAH;
46  2           END;

47  1           OUTPUT  (STATUS$2)    =DMA$REG$T;
                /*  WAIT FOR TC=0  */

48  1               DO WHILE   (INPUT   (CMD$37)  AND TC) <>  TC;
49  2               END;

50  1           GOTO START91;

51  1      END;


MODULE INFORMATION

     CODE AREA SIZE      =0122H    290D
     CONSTANT AREA SIZE  =0000H      0D
     VARIABLE AREA SIZE  =0001H      1D
     MAXIMUM STACK SIZE  =0000H      0D
     152 LINES READ
     0 PROGRAM ERROR (S)
```

230832-12

3

APPENDIX C
SOFTWARE FOR HP 9835A

```
10   REM SEND IN
TERFACE CLEAR
20   ABORTIO 7
30   REM FORCE E
RRORS UNTIL LIST
ENERS ACTIVE
40 Frcerr:   OUT
PUT 704 USING "#
,K";"B"
50 Chkstst:   ST
ATUS 7;Stat1,Sta
t2,Stat3,Stat4
60   Err=Stat2 A
ND 1
70   IF Err=1 TH
EN GOTO Frcerr
80   PRINT CHR$(
12),"LISTENERS A
RE ON LINE "
90   REM CONFIGU
RE PPOLL
100  PPOLL CONFI
GURE 704;"0000C1
00"
110

   ! response on
 bit 4
120  PRINT CHR$(
12),"PARALLEL PO
LL CONFIGURED"
130  REM ENABLE
KEYBOARD INTERRU
PT
140  PRINT "COMM
AND = ?     (HIT
 'H' FOR LIST)"
150 Keyen:   ON K
BD GOSUB 610
160  STATUS 7;St
at1,Stat2,Stat3,
Stat4
170  Sra=BINAND(
Stat1,128)
180  IF Sra=0 TH
EN GOTO Keyen
190  OFF KBD
200  PRINT CHR$(
12),"SRQ RECEIVE
D"
210  PRINT "SEND
ING PARALLEL POL
L RESPONSE MESSA
GE"
220  REM EXECUTI
NG PARALLEL POLL
230  Ppollbyte=P
POLL(7)
240  PRINT "PARA
LLEL POLL BYTE =
 ";Ppollbyte
250  PRINT "----
----------
------"
260  Ppollbyte=B
INAND(Ppollbyte,
8)
270  IF Ppollbyt
e=0 THEN GOTO P8
291
280  PRINT "SR
NOT FROM 8291"
281  PRINT "COMM
AND = ?     (HIT
'H' FOR LIST)"
290  GOTO Keyen
300 P8291:   PRIN
T "SRQ IS FROM N
CC 8291 ... THE
ENTERPRISE"
310  PRINT "PERF
ORMING SERIAL PO
LL TO GET STATUS
 "
320  STATUS 704;
Stat
330  PRINT CHR$(
12),"Status = ";
Stat
340  Dxfer=BINAN
D(Stat,1)
520  IF Dxfer>0
THEN GOTO Rcvr
530  GOTO Keyen
531 Rcvr:   REM R
EADY TO RCV CHAR
S FROM GPIB
540  DIM G$[80]
550  ENTER 704 U
SING "%,T";G$
560  PRINT CHR$(
12),G$
570  PRINT "COMM
AND = ?     (HIT
 'H' FOR LIST)"
580  GOTO Keyen
590  REM INTERRU
PT SERVICE ROUTI
NES
600  REM GET KEY
BOARD DATA
610 Whatkey:   DI
M K$[80]
620  K$=KBD$
630  IF K$="G" T
HEN GOTO Get
640  IF K$="D" T
HEN GOTO Dec
650  IF K$="R" T
HEN GOTO Rem
660  IF K$="H" T
HEN GOTO Help
670  IF K$="X" T
HEN GOTO Xmit
680 Get:   TRIGGE
R 704
690  PRINT CHR$(
12),"GROUP EXECU
TE TRIGGER SENT"
700  PRINT " "
710  PRINT "COMM
AND = ?     (HIT
'H' FOR LIST)"
720  RETURN
730 Dec:   RESET
704
740  PRINT CHR$(
12),"SELECTIVE D
EVICE CLEAR SENT
"
750  PRINT " "
760  PRINT "COMM
AND = ?     (HIT
'H' FOR LIST)"
770  RETURN
780 Rem:   LOCAL
704
790  PRINT CHR$(
12),"REMOTE MESS
AGE SENT"
800  PRINT " "
810  PRINT "COMM
AND = ?     (HIT
 'H' FOR LIST)"
820  RETURN
830 Help:   PRINT
 CHR$(12)
840  PRINT " @@@
@ OPERATOR ALLOW
ABLE COMMANDS @@
@@ "
850  PRINT " hit
 key    result"
860  PRINT "    G
      Send GET m
essage"
870  PRINT "    D
      Send DEC m
essage"
880  PRINT "    R
      Send REM L
OC message"
890  PRINT "    X
      Xmits keyb
oard input to 82
91"
900  PRINT "    H
      Prints thi
s table"
910  PRINT " "
920  PRINT "...
go ahead, TRY IT
!"
930  RETURN
940 Xmit:   DIM A
$[80]
950  PRINT CHR$(
12),"Enter data
to send and hit
CONTINUE"
960  INPUT A$
970  OUTPUT 704;
A$
971  EOI 7;0
980  PRINT "COMM
AND = ?     (HIT
 'H' FOR LIST)"
990  RETURN
1000 END
```

230832-13

230832-14

230832-15

APPENDIX D
SOFTWARE FOR HP 8088/HP 9835A VIA GPIB

```
PL/M-86 COMPILER     HPIB

ISIS-II PL/M-86 V1.1 COMPILATION OF MODULE HPIB
OBJECT MODULE PLACED IN :F1:HPIB.OBJ
COMPILER INVOKED BY:  PLM86 :F1:HPIB.SRC LARGE


   1          HPIB:
              /*

              PARAMETER DECLARATIONS
              */

              DO;

   2   1      DECLARE

              ADDR$HI       LITERALLY   '01H',
              ADDR$LO       LITERALLY   '00H',
              ADSC          LITERALLY   '01H',
              BI        LITERALLY   '01H',
              BO        LITERALLY   '02H',
              CHAR$COUNT    BYTE,
              CHAR          BYTE,
              CHARS(80)     BYTE,
              CLEAR         LITERALLY   '00H',
              CPT       LITERALLY   '80H',
              CRLF          LITERALLY   '0AH',
              DEC       LITERALLY   '08H',
              DMA$ADR$LSTN      POINTER,
              DMA$ADR$TALK      POINTER,
              DMA$WRD$LSTN(2) WORD AT       (@DMA$ADR$LSTN),
              DMA$WRD$TALK(2) WORD AT       (@DMA$ADR$TALK),
              DMA$REG$L     LITERALLY   '10H',
              DMA$REG$T     LITERALLY   '20H',
              DNE       LITERALLY   '10H',
              END$EOI       LITERALLY   '88H',
              EOS       LITERALLY   '0DH',
              ERR       LITERALLY   '04H',
              GET       LITERALLY   '20H',
              I         BYTE,
              LISTEN        LITERALLY   '04H',
              MLA       LITERALLY   '04H',
              MODE$1        LITERALLY   '01H',
              NO$DMA        LITERALLY   '00H',
              NO$RSV        LITERALLY   '00H',
              NORM$TIME     LITERALLY   '20H',
              PON       LITERALLY   '00H',
              PPC       LITERALLY   '05H',
              PPE$MASK      LITERALLY   '60H',
              PPOLL$CNFG$FLAG LITERALLY     '01H',
              PPOLL$EN$BYTE     BYTE,
              PRI$BUF(80) BYTE AT       (@CHARS),
              RD$XFER       LITERALLY   '48H',
              RESET         LITERALLY   '02H',
              REMC          LITERALLY   '02H',
              RSV       LITERALLY   '40H',
              RXRDY         LITERALLY   '02H',
```

230832-16

```
3   1       DECLARE

                ADDR$O          LITERALLY   'OFFC6H',
                ADDR$STATUS     LITERALLY   'OFFC4H',
                CLEAR$FF        LITERALLY   'OFFDDH',
                CMD$37          LITERALLY   'OFFD8H',
                COMMAND$MOD     LITERALLY   'OFFC5H',
                COUNT$HI        LITERALLY   'OFFD1H',
                COUNT$LO        LITERALLY   'OFFD1H',
                CPT$REQ         LITERALLY   'OFFC5H',
                EOS$REQ         LITERALLY   'OFFC7H',
                PORT$IN         LITERALLY   'OFFC0H',
                PORT$OUT        LITERALLY   'OFFC0H',
                SER$DATA        LITERALLY   'OFFF0H',
                SER$STAT        LITERALLY   'OFFF2H',
                SET$MASK        LITERALLY   'OFFDFH',
                SET$MODE        LITERALLY   'OFFDBH',
                SPOLL$STAT      LITERALLY   'OFFC3H',
                START$HI        LITERALLY   'OFFD0H',
                START$LO        LITERALLY   'OFFD0H',
                STATUS$1        LITERALLY   'OFFC1H',
                STATUS$2        LITERALLY   'OFFC2H';

                /* crt  messages list */


4   1       DECLARE GET$MSG(11) BYTE DATA (0DH,0AH,'TRIGGER',0AH,0DH);
5   1       DECLARE DEC$MSG(16) BYTE DATA (0DH,0AH,'DEVICE CLEAR',0AH,0DH);
6   1       DECLARE REMC$MSG(10) BYTE DATA (0DH,0AH,'REMOTE',0DH,0AH);
7   1       DECLARE CPT$MSG(22) BYTE DATA (0DH,0AH,'UNDEF CMD RECEIVED',0AH,0DH);
8   1       DECLARE HUH$MSG(11) BYTE DATA (0DH,0AH,'HUH ???',0DH,0AH);


                    /* called procedures */

9   1       REQSER:     PROCEDURE;
```

230832-17

```
PL/M-86 COMPILER    HPIB

  10   2              OUTPUT (SPOLL$STAT)=TRQ;

  11   2              DO WHILE (INPUT (SPOLL$STAT) AND SRQS)=SRQS;
  12   3              END;

  13   2              OUTPUT (SPOLL$STAT)=NO$RSV;

  14   2              END REQSER;

  15   1      CO: PROCEDURE(XXX);
  16   2              DECLARE
                       XXX         BYTE;

  17   2              DO WHILE (INPUT (SER$STAT) AND TXRDY)<>TXRDY;
  18   3              END;
  19   2              OUTPUT (SER$DATA)=XXX;
  20   2          END CO;
  21   1      HUH:        PROCEDURE;
  22   2                  DO I=0 TO 10;
  23   3                     CALL CO (HUH$MSG(I));
  24   3                  END;
  25   2              END HUH;

  26   1      CI:     PROCEDURE;

  27   2              IF (INPUT (SER$STAT) AND RXRDY)=RXRDY THEN
  28   2                  DO;
  29   3                      I=0;
  30   3                      CHAR$COUNT=0;
  31   3      STORE$CHAR:         CHAR=(INPUT (SER$DATA) AND 7FH);
  32   3                      CHAR$COUNT=CHAR$COUNT+1;
  33   3                      CALL CO (CHAR);
  34   3                      CHARS(I)=CHAR;
  35   3                      I=I+1;
  36   3                      IF CHAR <> CRLF THEN
  37   3                         DO;
  38   4                           DO WHILE (INPUT (SER$STAT) AND RXRDY) <>RXRDY;
  39   5                           END;
  40   4                           GOTO STORE$CHAR;
  41   4                         END;
  42   3                      CALL REQSER;
  43   3                  END;
  44   2              END CI;

  45   1      TALK$EXEC:  PROCEDURE;

  46   2              OUTPUT (STATUS$2)=CLEAR;

                      /*
                      manipulate address bits for DMA controller
                      */

  47   2              DMA$ADR$TALK=(@CHARS);
  48   2              DMA$WRD$TALK(1)=SHL(DMA$WRD$TALK(1),4);
  49   2              DMA$WRD$TALK(0)=DMA$WRD$TALK(0)+DMA$WRD$TALK(1);

  50   2              OUTPUT (CLEAR$FF)=CLEAR;
```

230832-18

3

```
PL/M-86 COMPILER    HPIB

  51   2              OUTPUT (CMD37)=NORM$TIME;
  52   2              OUTPUT (SET$MODE)=RD$XFER;
  53   2              OUTPUT (SET$MASK)=CLEAR;
  54   2              OUTPUT (START$LO)=DMA$WRD$TALK(0);
  55   2              DMA$WRD$TALK(0)=SHR(DMA$WRD$TALK(0),8);
  56   2              OUTPUT (START$HI)=DMA$WRD$TALK(0);
  57   2              OUTPUT (COUNT$LO)=CHAR$COUNT;
  58   2              OUTPUT (COUNT$HI)=0;

  59   2              OUTPUT (EOS$REG)=EOS;
  60   2              OUTPUT (COMMAND$MOD)=END$EOI;

  61   2              DO WHILE (INPUT (STATUS$1) AND BO)=0;
  62   3              END;
  63   2              OUTPUT (PORT$OUT)=0AAH;

  64   2              DO WHILE (INPUT (STATUS$1) AND ERR)=ERR;
  65   3                  DO WHILE (INPUT (STATUS$1) AND BO)=0;
  66   4                  END;
  67   3                  OUTPUT (PORT$OUT)=0AAH;
  68   3              END;
  69   2              OUTPUT (STATUS$2)=DMA$REQ$T;

  70   2              END TALK$EXEC;

  71   1    LISTEN$EXEC:    PROCEDURE;

  72   2              OUTPUT (STATUS$2)=CLEAR;
  73   2              OUTPUT (CLEAR$FF)=CLEAR;
  74   2              OUTPUT (CMD$37)=NORM$TIME;
  75   2              OUTPUT (SET$MODE)=WR$XFER;
  76   2              OUTPUT (SET$MASK)=CLEAR;
  77   2              DMA$ADR$LSTN=(@CHARS);
  78   2              DMA$WRD$LSTN(1)=SHL(DMA$WRD$LSTN(1),4);
  79   2              DMA$WRD$LSTN(0)=DMA$WRD$LSTN(0)+DMA$WRD$LSTN(1);
  80   2              OUTPUT (START$LO)=DMA$WRD$LSTN(0);
  81   2              DMA$WRD$LSTN(0)=SHR(DMA$WRD$LSTN(0),8);
  82   2              OUTPUT (START$HI)=DMA$WRD$LSTN(0);
  83   2              OUTPUT (COUNT$LO)=TC$LO;
  84   2              OUTPUT (COUNT$HI)=TC$HI;
  85   2              OUTPUT (STATUS$2)=DMA$REQ$L;

  86   2              END LISTEN$EXEC;

  87   1    PRINTER:    PROCEDURE;

  88   2              I=0;

  89   2              DO WHILE PRI$BUF(I) <>CRLF;
  90   3                  CALL CO (PRI$BUF(I));
  91   3                  I=I+1;
  92   3              END;
  93   2              CALL CO (PRI$BUF(I));

  94   2              END PRINTER;
```

230832-19

```
PL/M-86 COMPILER    HPIB


 95   1      ADSC$EXEC:  PROCEDURE;

 96   2              TA$OR$LA=INPUT (ADDR$STATUS);

 97   2              IF (TA$OR$LA AND TALK)=TALK THEN
 98   2                  CALL TALK$EXEC;
 99   2              IF (TA$OR$LA AND LISTEN)=LISTEN THEN
100   2                  CALL LISTEN$EXEC;

101   2              END ADSC$EXEC;

102   1      GET$EXEC:   PROCEDURE;
103   2                  DO I=0 TO 10;
104   3                   CALL CO (GET$MSG(I));
105   3                  END;
106   2              END GET$EXEC;

107   1      DEC$EXEC:   PROCEDURE;
108   2                  DO I=0 TO 15;
109   3                   CALL CO (DEC$MSG(I));
110   3                  END;
111   2              END DEC$EXEC;

112   1      REMC$EXEC:  PROCEDURE;
113   2                  DO I=0 TO 9;
114   3                   CALL CO (REMC$MSG(I));
115   3                  END;
116   2              END REMC$EXEC;

117   1      PPOLL$CON:  PROCEDURE;

118   2              OUTPUT (COMMAND$MOD)=PPOLL$CNFG$FLAG;

119   2              END PPOLL$CON;

120   1      PPOLL$EN:   PROCEDURE;

121   2              PPOLL$EN$BYTE=(UDC AND 6FH);
122   2              OUTPUT (COMMAND$MOD)=PPOLL$EN$BYTE;

123   2              END PPOLL$EN;

124   1      CPT$EXEC:   PROCEDURE;
125   2                  DO I=0 TO 21;
126   3                   CALL CO (CPT$MSG(I));
127   3                  END;

128   2                  UDC=INPUT (CPT$REG);
129   2                  UDC=(UDC AND 7FH);
130   2                  IF (UDC AND PPC)=PPC THEN
131   2                   CALL PPOLL$CON;

132   2                  IF (UDC AND PPE$MASK)=PPE$MASK THEN
133   2                   CALL PPOLL$EN;
```

230832-20

3

```
                    /* response to listeners check */


                    DO WHILE (INPUT (STATUS$1) AND BI)=0;
    142   2         END;

    143   1         XYZ=INPUT (PORT$IN);
    144   1         XYZ=INPUT (STATUS$2);

    145   1   CMD:

              RDSTAT:
                    /* read status registers and interpret command */

                    STAT1=INPUT (STATUS$1);
    146   1         STAT2=INPUT (STATUS$2);

    147   1         IF (STAT1 AND DNE)=DNE THEN
    148   1            CALL PRINTER;
    149   1         IF (STAT1 AND CPT)=CPT THEN
    150   1             DO;
    151   2             CALL CPT$EXEC;
    152   2             STAT2=(STAT2 AND OFEH);
    153   2             END;
    154   1         IF (STAT1 AND GET)=GET THEN
    155   1             DO;
    156   2             CALL GET$EXEC;
    157   2             STAT2=(STAT2 AND OFEH);
    158   2             END;
    159   1         IF (STAT1 AND DEC)=DEC THEN
    160   1             DO;
    161   2             CALL DEC$EXEC;
    162   2             STAT2=(STAT2 AND OFEH);
    163   2             END;
    164   1         IF (STAT2 AND REMC)=REMC THEN
    165   1             DO;
    166   2             CALL REMC$EXEC;
    167   2             STAT2=(STAT2 AND OFEH);
    168   2             END;
    169   1         IF (STAT2 AND ADSC)=ADSC THEN
```

230832-21

```
PL/M-86 COMPILER     HPIB
                       .

 170   1                 DO;
 171   2                 CALL ADSC$EXEC;
 172   2                 STAT2=(STAT2 AND OFEH);
 173   2                 END;

 174   1           CALL CI;

 175   1           GOTO CMD;



 176   1       END;


MODULE INFORMATION:

     CODE AREA SIZE     = 0475H   1141D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 0061H     97D
     MAXIMUM STACK SIZE = 000AH     10D
     349 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

230832-22

3

June 1989

Using the 8292 GPIB Controller

Order Number: 231324-002

USING THE 8292 GPIB CONTROLLER

CONTENTS

PAGE

INTRODUCTION 3-98

GPIB/IEEE 488 OVERVIEW 3-98

HARDWARE ASPECTS OF THE SYSTEM 3-104

8291 Talker/Listener 3-104

8292 Controller 3-105

8293 Bus Transceivers 3-105

ZT7488/18 GPIB Controller 3-108

8292 COMMAND DESCRIPTION 3-110

SOFTWARE DRIVER OUTLINE 3-114

Initialization 3-115

Talker/Listener 3-115

Send Data 3-115

Receive Data 3-117

Transfer Data 3-119

Controller 3-119

Trigger 3-119

Device Clear 3-120

Serial Poll 3-120

Parallel Poll 3-121

Pass Control 3-123

Receive Control 3-124

Service Request 3-126

System Controller 3-126

Remote 3-126

Local 3-126

Interface Clear/Abort 3-127

INTERRUPT AND DMA CONSIDERATIONS 3-127

APPLICATION EXAMPLE 3-128

CONCLUSION 3-129

APPENDIX A 3-130

Source Listings 3-130–3-147

APPENDIX B 3-148

Test Cases for the Software Drivers 3-148

APPENDIX C 3-154

Remote Message Coding 3-154

INTRODUCTION

The Intel® 8292 is a preprogrammed UPI™-41A that implements the Controller function of the IEEE Std 488-1978 (GPIB, HP-IB, IEC Bus, etc.). In order to function the 8292 must be used with the 8291 Talker/Listener and suitable interface and transceiver logic such as a pair of Intel 8293s. In this configuration the system has the potential to be a complete GPIB Controller when driven by the appropriate software. It has the following capabilities: System Controller, send IFC and Take Charge, send REN, Respond to SRQ, send Interface messages, Receive Control, Pass Control, Parallel Poll and Take Control Synchronously.

This application note will explain the 8292 only in the system context of an 8292, 8291, two 8293s and the driver software. If the reader wishes to learn more about the UPI-41A aspects of the 8292, Intel's Application Note AP-41 describes the hardware features and programming characteristics of the device. Additional information on the 8291 may be obtained in the data sheet. The 2893 is detailed in its data sheet. Both chips will be covered here in the details that relate to the GPIB controller.

The next section of this application note presents an overview of the GPIB in a tutorial, but comprehensive nature. The knowledgeable reader may wish to skip this section; however, certain basic semantic concepts introduced there will be used throughout this note.

Additional sections cover the view of the 8292 from the CPU's data bus, the interaction of the 3 chip types (8291, 8292, 8293), the 8292's software protocol and the system level hardware/software protocol. A brief description of interrupts and DMA will be followed by an application example. Appendix A contains the source code for the system driver software.

GPIB/IEEE 488 OVERVIEW

Design Objectives

WHAT IS THE IEEE 488 (GPIB)?

The experience of designing systems for a variety of applications in the early 1970's caused Hewlett-Packard to define a standard intercommunication mechanism which would allow them to easily assemble instrumentation systems of varying degrees of complexity. In a typical situation each instrument designer designed his/her own interface from scratch. Each one was inconsistent in terms of electrical levels, pin-outs on a connector, and types of connectors. Every time they built a system they had to invent new cables and new documentation just to specify the cabling and interconnection procedures.

Based on this experience, Hewlett-Packard began to define a new interconnection scheme. They went further than that, however, for they wanted to specify the typical communication protocol for systems of instruments. So in 1972, Hewlett-Packard came out with the first version of the bus which since has been modified and standardized by a committee of several manufacturers, coordinated through the IEEE, to perfect what is now known as the IEEE 488 Interface Bus (also known as the HPIB, the GPIB and the IEC bus). While this bus specification may not be perfect, it is a good compromise of the various desires and goals of instrumentation and computer peripheral manufacturers to produce a common interconnection mechanism. It fits most instrumentation systems in use today and also fits very well the microcomputer I/O bus requirements. The basic design objectives for the GPIB were to:

1) Specify a system that is easy to use, but has all of the terminology and the definitions related to that system precisely spelled out so that everyone uses the same language when discussing the GPIB.
2) Define all of the mechanical, electrical, and functional interface requirements of a system, yet not define any of the device aspects (they are left up to the instrument designer).
3) Permit a wide range of capabilities of instruments and computer peripherals to use a system simultaneously and not degrade each other's performance.
4) Allow different manufacturers' equipment to be connected together and work together on the same bus.
5) Define a system that is good for limited distance interconnections.
6) Define a system with minimum restrictions on performance of the devices.
7) Define a bus that allows asynchronous communication with a wide range of data rates.
8) Define a low cost system that does not require extensive and elaborate interface logic for the low cost instruments, yet provides higher capability for the higher cost instruments if desired.
9) Allow systems to exist that do not need a central controller; that is, communication directly from one instrument to another is possible.

Although the GPIB was originally designed for instrumentation systems, it became obvious that most of these systems would be controlled by a calculator or computer. With this in mind several modifications were made to the original proposal before its final adoption as an international standard. Figure 1 lists the salient characteristics of the GPIB as both an instrumentation bus and as a computer I/O bus.

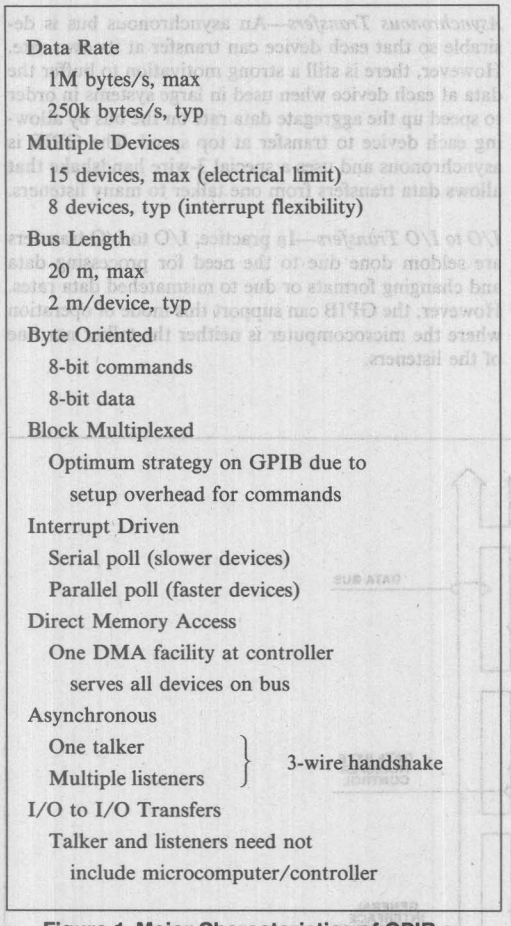


Figure 1. Major Characteristics of GPIB as Microcomputer I/O Bus

The bus can be best understood by examining each of these characteristics from the viewpoint of a general microcomputer I/O bus.

Data Rate—Most microcomputer systems utilize peripherals of differing operational rates, such as floppy discs at 31k or 62k bytes/s (single or double density), tape cassettes at 5k to 10k bytes/s, and cartridge tapes at 40k to 80k bytes/s. In general, the only devices that need high speed I/O are 0.5″ (1.3-cm) magnetic tapes and hard discs, operational at 30k to 781k bytes/s, respectively. Certainly, the 250k-bytes/s data rate that can be easily achieved by the IEEE 488 bus is sufficient for microcomputers and their peripherals, and is more than needed for typical analog instruments that take only a few readings per second. The 1M-byte/s maximum data rate is not easily achieved on the GPIB and requires special attention to considerations beyond the scope of this note. Although not required, data buffering in each device will improve the overall bus performance and allow utilization of more of the bus bandwidth.

Multiple Devices—Many microcomputer systems used as computers (not as components) service from three to seven peripherals. With the GPIB, up to 8 devices can be handled easily by 1 controller; with some slowdown in interrupt handling, up to 15 devices can work together. The limit of 8 is imposed by the number of unique parallel poll responses available; the limit of 15 is set by the electrical drive characteristics of the bus. Logically, the IEEE 488 Standard is capable of accommodating more device addresses (31 primary, each potentially with 31 secondaries).

Bus Length—Physically, the majority of microcomputer systems fit easily on a desk top or in a standard 19″ (48-cm) rack, eliminating the need for extra long cables. The GPIB is designed typically to have 2m of length per device, which accommodates most systems. A line printer might require greater cable lengths, but this can be handled at the lower speeds involved by using extra dummy terminations.

Byte Oriented—The 8-bit byte is almost universal in I/O applications; even 16-bit and 32-bit computers use byte transfers for most peripherals. The 8-bit byte matches the ASCII code for characters and is an integral submultiple of most computer word sizes. The GPIB has an 8-bit wide data path that may be used to transfer ASCII or binary data, as well as the necessary status and control bytes.

Block Multiplexed—Many peripherals are block oriented or are used in a block mode. Bytes are transferred in a fixed or variable length group; then there is a wait before another group is sent to that device, e.g., one sector of a floppy disc, one line on a printer or type punch, etc. The GPIB is, by nature, a block multiplexed bus due to the overhead involved in addressing various devices to talk and listen. This overhead is less bothersome if it only occurs once for a large number of data bytes (once per block). This mode of operation matches the needs of microcomputers and most of their peripherals. Because of block multiplexing, the bus works best with buffered memory devices.

Interrupt Driven—Many types of interrupt systems exist, ranging from complex, fast, vectored/priority networks to simple polling schemes. The main tradeoff is usually cost versus speed of response. The GPIB has two interrupt protocols to help span the range of applications. The first is a single service request (SRQ) line that may be asserted by all interrupting devices. The controller then polls all devices to find out which wants service. The polling mechanism is well defined and can

be easily automated. For higher performance, the parallel poll capability in the IEEE 488 allows up to eight devices to be polled at once—each device is assigned to one bit of the data bus. This mechanism provides fast recognition of an interrupting device. A drawback is the frequent need for the controller to explicitly conduct a parallel poll, since there is no equivalent of the SRQ line for this mode.

Direct Memory Access (DMA)—In many applications, no immediate processing of I/O data on a byte-by-byte basis is needed or wanted. In fact, programmed transfers slow down the data transfer rate unnecessarily in these cases, and higher speed can be obtained using DMA. With the GPIB, one DMA facility at the controller serves all devices. There is no need to incorporate complex logic in each device.

Asynchronous Transfers—An asynchronous bus is desirable so that each device can transfer at its own rate. However, there is still a strong motivation to buffer the data at each device when used in large systems in order to speed up the aggregate data rate on the bus by allowing each device to transfer at top speed. The GPIB is asynchronous and uses a special 3-wire handshake that allows data transfers from one talker to many listeners.

I/O to I/O Transfers—In practice, I/O to I/O transfers are seldom done due to the need for processing data and changing formats or due to mismatched data rates. However, the GPIB can support this mode of operation where the microcomputer is neither the talker nor one of the listeners.

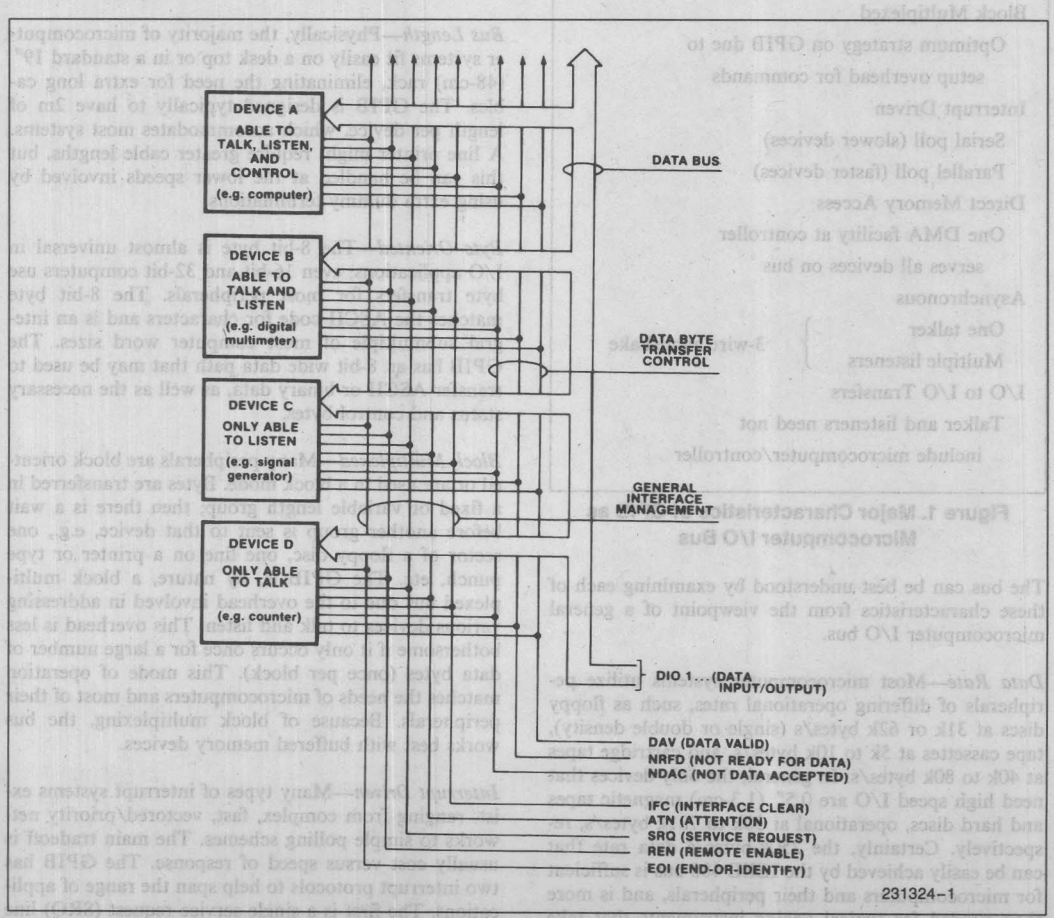


Figure 2. Interface Capabilities and Bus Structure

GPIB Signal Lines

DATA BUS

The lines DI01 through DI08 are used to transfer addresses, control information and data. The formats for addresses and control bytes are defined by the IEEE 488 standard (see Appendix C). Data formats are undefined and may be ASCII (with or without parity) or binary. D101 is the Least Significant bit (note that this will correspond to bit 0 on most computers).

MANAGEMENT BUS

ATN—Attention. This signal is asserted by the Controller to indicate that it is placing an address or control byte on the Data Bus. ATN is de-asserted to allow the assigned Talker to place status or data on the Data Bus. The Controller regains control by reasserting ATN; this is normally done synchronously with the handshake to avoid confusion between control and data bytes.

EOI—End or Identify. This signal has two uses as its name implies. A talker may assert EOI simultaneously with the last byte of data to indicate end of data. The Controller may assert EOI along with ATN to initiate a Parallel Poll. Although many devices do not use Parallel Poll, all devices *should* use EOI to end transfers (many currently available ones do not).

SRQ—Service Request. This line is like an interrupt: it may be asserted by any device to request the Controller to take some action. The Controller must determine which device is asserting SRQ by conducting a Serial Poll at its earliest convenience. The device deasserts SRQ when polled.

IFC—Interface Clear. This signal is asserted only by the System Controller in order to initialize all device interfaces to a known state. After deasserting IFC, the System Controller is the active controller of the system.

REN—Remote Enable. This signal is asserted only by the System Controller. Its assertion does not place devices into Remote Control mode; REN only *enables* a device to go remote when addressed to listen. When in Remote, a device should ignore its front panel controls.

TRANSFER BUS

NRFD—Not Ready For Data. This handshake line is asserted by a listener to indicate it is not yet ready for the next data or control byte. Note that the Controller will not see NRFD deasserted (i.e., ready for data) until all devices have deasserted NRFD.

NDAC—Not Data Accepted. This handshake line is asserted by a Listener to indicate it has not yet accepted the data or control byte on the DIO lines. Note that the Controller will not see NDAC deasserted (i.e., data accepted) until all devices have deasserted NDAC.

DAV—Data Valid. This handshake line is asserted by the Talker to indicate that a data or control byte has been placed on the DIO lines and has had the minimum specified settling time.

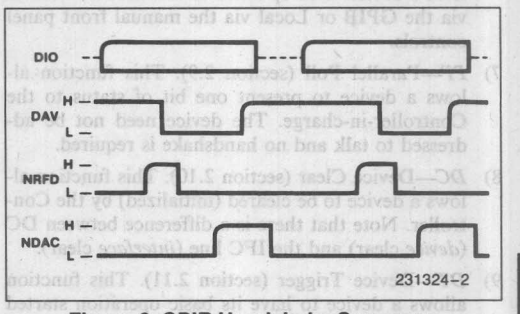


Figure 3. GPIB Handshake Sequence

GPIB Interface Functions

There are ten (10) interface functions specified by the IEEE 488 standard. Not all devices will have all functions and some may only have partial subsets. The ten functions are summarized below with the relevant section number from the IEEE document given at the beginning of each paragraph. For further information please see the IEEE standard.

1) *SH*—Source Handshake (section 2.3). This function provides a device with the ability to properly transfer data from a Talker to one or more Listeners using the three handshake lines.
2) *AH*—Acceptor Handshake (section 2.4). This function provides a device with the ability to properly receive data from the Talker using the three handshake lines. The AH function may also delay the beginning (NRFD) or end (NDAC) of any transfer.
3) *T*—Talker (section 2.5). This function allows a device to send status and data bytes when addressed to talk. An address consists of one (Primary) or two (Primary and Secondary) bytes. The latter is called an extended Talker.

3

addressed to talk and no handshake is required.

8) *DC*—Device Clear (section 2.10). This function allows a device to be cleared (initialized) by the Controller. Note that there is a difference between DC (*device* clear) and the IFC line (*interface* clear).

9) *DT*—Device Trigger (section 2.11). This function allows a device to have its basic operation started either individually or as part of a group. This capability is often used to synchronize several instruments.

10) *C*—Controller (section 2.12). This function allows a device to send addresses, as well as universal and addressed commands to other devices. There may be more than one controller on a system, but only one may be the controller-in-charge at any one time.

At power-on time the controller that is hardwired to be the System Controller becomes the active controller-in-charge. The System Controller has several unique capabilities including the ability to send Interface Clear (IFC—clears all device interfaces and returns control to the System Controller) and to send Remote Enable (REN—allows devices to respond to bus data once they are addressed to listen). The System Controller may optionally Pass Control to another controller, if the system software has the capability to do so.

GPIB Connector

The GPIB connector is a standard 24-pin industrial connector such as Cinch or Amphenol series 57 Micro-Ribbon. The IEEE standard specifies this connector, as well as the signal connections and the mounting hardware.

The cable has 16 signal lines and 8 ground lines. The maximum length is 20 meters with no more than two meters per device.

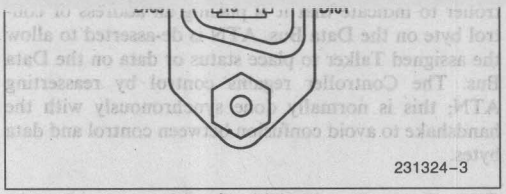


Figure 4. GPIB Connector

GPIB Signal Levels

The GPIB signals are all TTL compatible, low true signals. A signal is asserted (true) when its electrical voltage is less than 0.5 volts and is deasserted (false) when it is greater than 2.4 volts. Be careful not to become confused with the two handshake signals, NRFD and NDAC which are also low true (i.e. > 0.5 volts implies the device is Not Ready For Data).

The Intel 8293 GPIB transceiver chips ensure that all relevant bus driver/receiver specifications are met. Detailed bus electrical specifications may be found in Section 3 of the IEEE Std 488-1978. The Standard is the ultimate reference for all GPIB questions.

GPIB Message Protocols

The GPIB is a very flexible communications medium and as such has many possible variations of protocols. To bring some order to the situation, this section will discuss a protocol similar to the one used by Ziatech's ZT80 GPIB controller for Intel's MULTIBUS™ computers. The ZT80 is a complete high-level interface processor that executes a set of high level instructions that map directly into GPIB actions. The sequences of commands, addresses and data for these instructions provide a good example of how to use the GPIB (additional information is available in the ZT80 Instruction Manual). The 'null' at the end of each instruction is for cosmetic use to remove previous information from the DIO lines.

DATA—Transfer a block of data from device A to devices B, C . . .

1) Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2) Universal Unlisten
3) Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   Device C Primary (Listen) Address
   etc.
4) First Data Byte
   Second Data Byte
   .
   .
   .
   Last Data Byte (EOI)
5) Null

TRIGR—Trigger devices A, B . . . to take action

1) Universal Unlisten
2) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Group Execute Trigger
4) Null

PSCTL—Pass control to device A

1) Device A Primary (Talk) Address
   Device A Secondary Address (if any)
2) Talk Control
3) Null

CLEAR—Clear all devices

1) Device Clear
2) Null

REMAL—Remote Enable

1) Assert REN continuously

GOREM—Put devices A, B, . . . into Remote

1) Assert REN continuously
2) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Null

GOLOC—Put devices A, B, . . . into Local

1) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
2) Go To Local
3) Null

LOCAL—Reset all devices to Local

1) Stop asserting REN

LLKAL—Prevent all devices from returning to Local

1) Local Lock Out
2) Null

SPOLL—Conducts a serial poll of devices A, B, . . .

1) Serial Poll Enable
2) Universal Unlisten
3) ZT 80 Primary (Listen) Address
   ZT 80 Secondary Address
4) Device Primary (Talk) Address
   Device Secondary Address (if any)
5) Status byte from device
6) Go to Step 4 until all devices on list have been polled
7) Serial Poll Disable
8) Null

PPUAL—Unconfigure and disable Parallel Poll response from all devices

1) Parallel Poll Unconfigure
2) Null

ENAPP—Enable Parallel Poll response in devices A, B, . . .

1) Universal Unlisten
2) Device Primary (Listen) Address
   Device Secondary Address (if any)
3) Parallel Poll Configure
4) Parallel Poll Enable
5) Go to Step 2 until all devices on list have been configured.
6) Null

DISPP—Disable Parallel Poll response from devices A, B, . . .

1) Universal Unlisten
2) Device A Primary (Listen) Address
   Device A Secondary Address (if any)
   Device B Primary (Listen) Address
   Device B Secondary Address (if any)
   etc.
3) Disable Parallel Poll
4) Null

This Ap Note will detail how to implement a useful subset of these controller instructions.

HARDWARE ASPECTS OF THE SYSTEM

8291 GPIB Talker/Listener

The 8291 is a custom designed chip that implements many of the non-controller GPIB functions. It provides hooks so the user's software can implement additional features to complete the set. This chip is discussed in detail in its data sheet. The major features are summarized here:

— Designed to interface microprocessors to the GPIB
— Complete Source and Acceptor Handshake
— Complete Talker and Listener Functions with extended addressing
— Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local functions
— Programmable data transfer rate
— Maskable interrupts
— On-chip primary and secondary address recognition
— 1–8 MHz clock range
— 16 registers (8 read, 8 write) for CPU interface
— DMA handshake provision
— Trigger output pin
— On-chip EOS (End of Sequence)

The pinouts and block diagram are shown in Figure 5. One of eight read registers is for data transfer to the CPU; the other seven allow the microprocessor to monitor the GPIB states and various bus and device conditions. One of the eight write registers is for data transfer from the CPU; the other seven control various features of the 8291.

The 8291 interface functions will be software configured in this application example to the following subsets for use with the 8292 as a controller that does not pass control. The 8291 is used only to provide the handshake logic and to send and receive data bytes. It is not acting as a normal device in this mode, as it never sees ATN asserted.

SH1	Source Handshake
AH1	Acceptor Handshake
T3	Basic Talk-Only
L1	Basic Listen-Only
SR0	No Service Requests
RL0	No Remote/Local
PP0	No Parallel Poll Response
DC0	No Device Clear
DT0	No Device Trigger

If control is passed to another controller, the 8291 must be reconfigured to act as a talker/listener with the following subsets:

SH1	Source Handshake
AH1	Acceptor Handshake
T5	Basic Talker and Serial Poll
L3	Basic Listener
SR1	Service Requests
RL1	Remote/Local with Lockout
PP2	Reconfigured Parallel Poll
DC1	Device Clear
DT1	Device Trigger
C0	Not a Controller

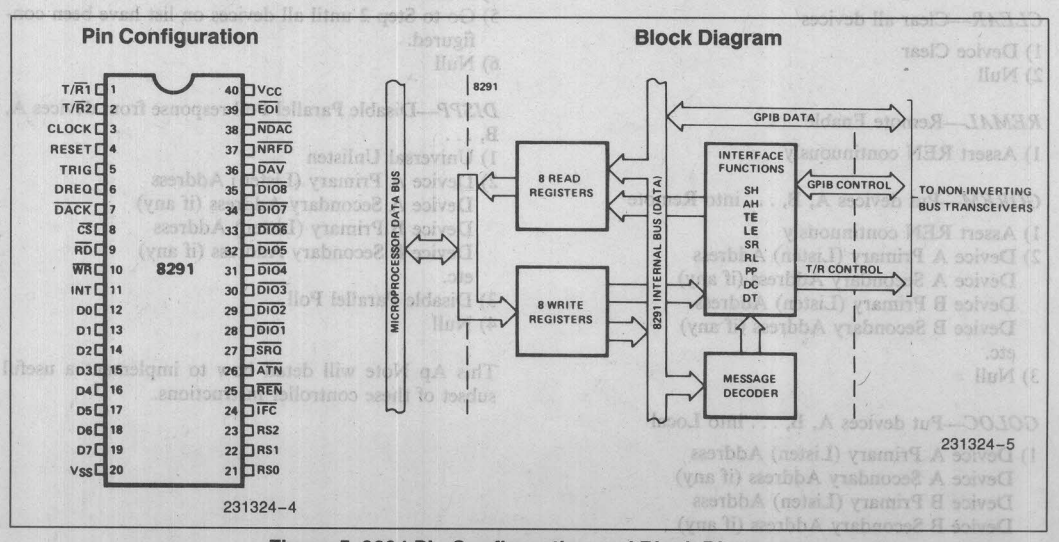


Figure 5. 8291 Pin Configuration and Block Diagram

Most applications do not pass control and the controller is always the system controller (see 8292 commands below).

8292 GPIB Controller

The 8292 is a preprogrammed Intel 8051A that provides the additional functions necessary to implement a GPIB controller when used with an 8291 Talker/Listener. The 8041A is documented in both a user's manual and in AP-41. The following description will serve only as an outline to guide the later discussion.

The 8292 acts as an intelligent slave processor to the main system CPU. It contains a processor, memory, I/O and is programmed to perform a variety of tasks associated with GPIB controller operation. The on-chip RAM is used to store information about the state of the Controller function, as well as a variety of local variables, the stack and certain user status information. The timer/counter may be optionally used for several time-out functions or for counting data bytes transferred. The I/O ports provide the GPIB control signals, as well as the ancillary lines necessary to make the 8291, 2, 3 work together.

The 8292 is closely coupled to the main CPU through three on-chip registers that may be independently accessed by both the master and the 8292 (UPI-41A). Figure 6 shows this Register Interface. Also refer to Figure 12.

The status register is used to pass Interrupt Status information to the master CPU (A0 = 1 on a read).

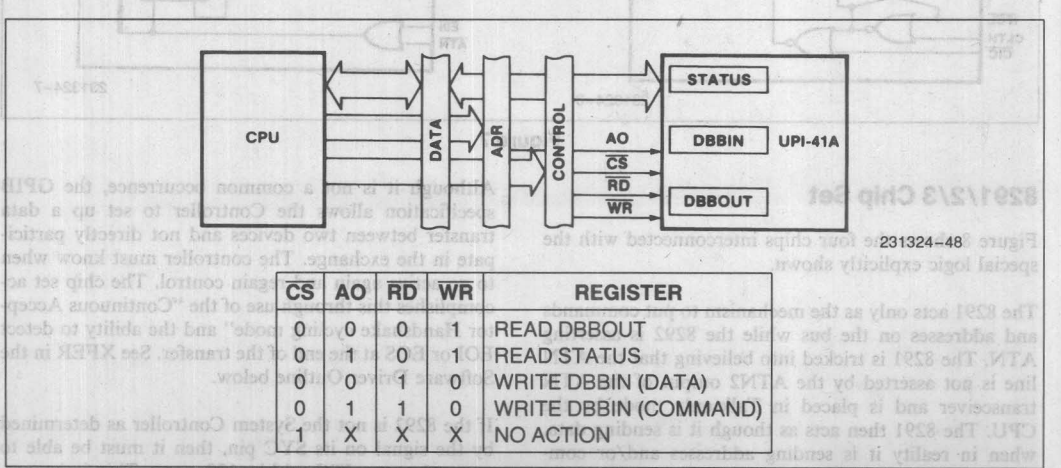
The DBBOUT register is used to pass one of five other status words to the master based on the last command written into DBBIN. DBBOUT is accessed when A0 = 0 on a Read. The five status words are Error Flag, Controller Status, GPIB Status, Event Counter Status or Time Out Status.

DBBIN receives either commands (A0 = 1 on a Write) or command related data (A0 = 0 on a write) from the master. These command related data are Interrupt Mask, Error Mask, Event Counter or Time Out.

8293 GPIB Transceivers

The 8293 is a multi-use HMOS chip that implements the IEEE 488 bus transceivers and contains the additional logic required to make the 8291 and 8292 work together. The two option strapping pins are used to internally configure the chip to perform the specialized gating required for use with 8291 as a device or with 8291/92 as a controller.

In this application example the two configurations used are shown in Figure 7a and 7b. The drivers are set to open collector or three state mode as required and the special logic is enabled as required in the two modes.



CS	AO	RD	WR	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION

Figure 6. UPI-41A Registers

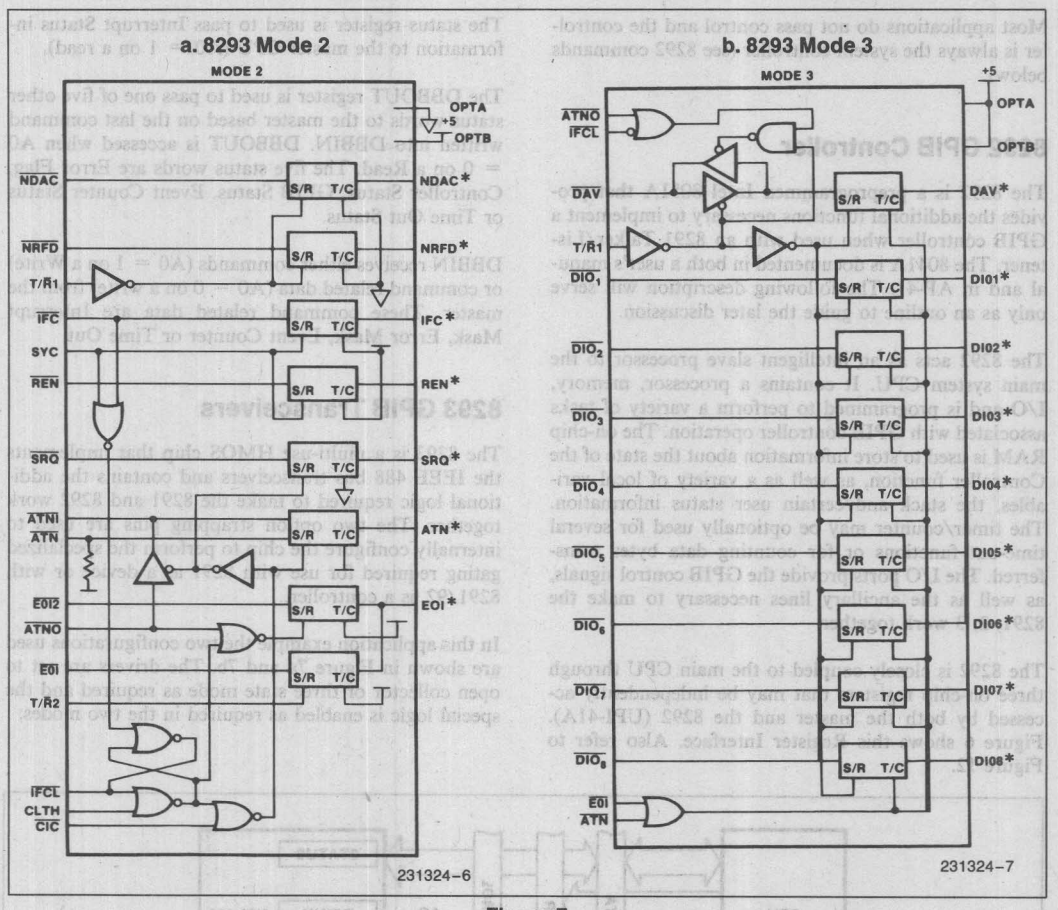


Figure 7

8291/2/3 Chip Set

Figure 8 shows the four chips interconnected with the special logic explicitly shown.

The 8291 acts only as the mechanism to put commands and addresses on the bus while the 8292 is asserting ATN. The 8291 is tricked into believing that the ATN line is not asserted by the ATN2 output of the ATN transceiver and is placed in Talk-only mode by the CPU. The 8291 then acts as though it is sending data, when in reality it is sending addresses and/or commands. When the 8292 deasserts ATN, the CPU software must place the 8291 in Talk-only, Listen-only or Idle based on the implicit knowledge of how the controller is going to participate in the data transfer. In other words, the 8291 does not respond directly to addresses or commands that it sends on the bus on behalf of the Controller. The user software, through the use of Listen-only or Talk-only, makes the 8291 behave as though it were addressed.

Although it is not a common occurrence, the GPIB specification allows the Controller to set up a data transfer between two devices and not directly participate in the exchange. The controller must know when to go active again and regain control. The chip set accomplishes this through use of the "Continuous Acceptor Handshake cycling mode" and the ability to detect EOI or EOS at the end of the transfer. See XFER in the Software Driver Outline below.

If the 8292 is not the System Controller as determined by the signal on its SYC pin, then it must be able to respond to an IFC within 100 μsec. This is accomplished by the cross-coupled NORs in Figure 7a which deassert the 8293's internal version of $\overline{CIC}$ (Not Controller-in-Charge). This condition is latched until the 8292's firmware has received the IFCL (interface clear received latch) signal by testing the $\overline{IFCL}$ input. The firmware then sets its signals to reflect the inactive condition and clears the 8293's latch.

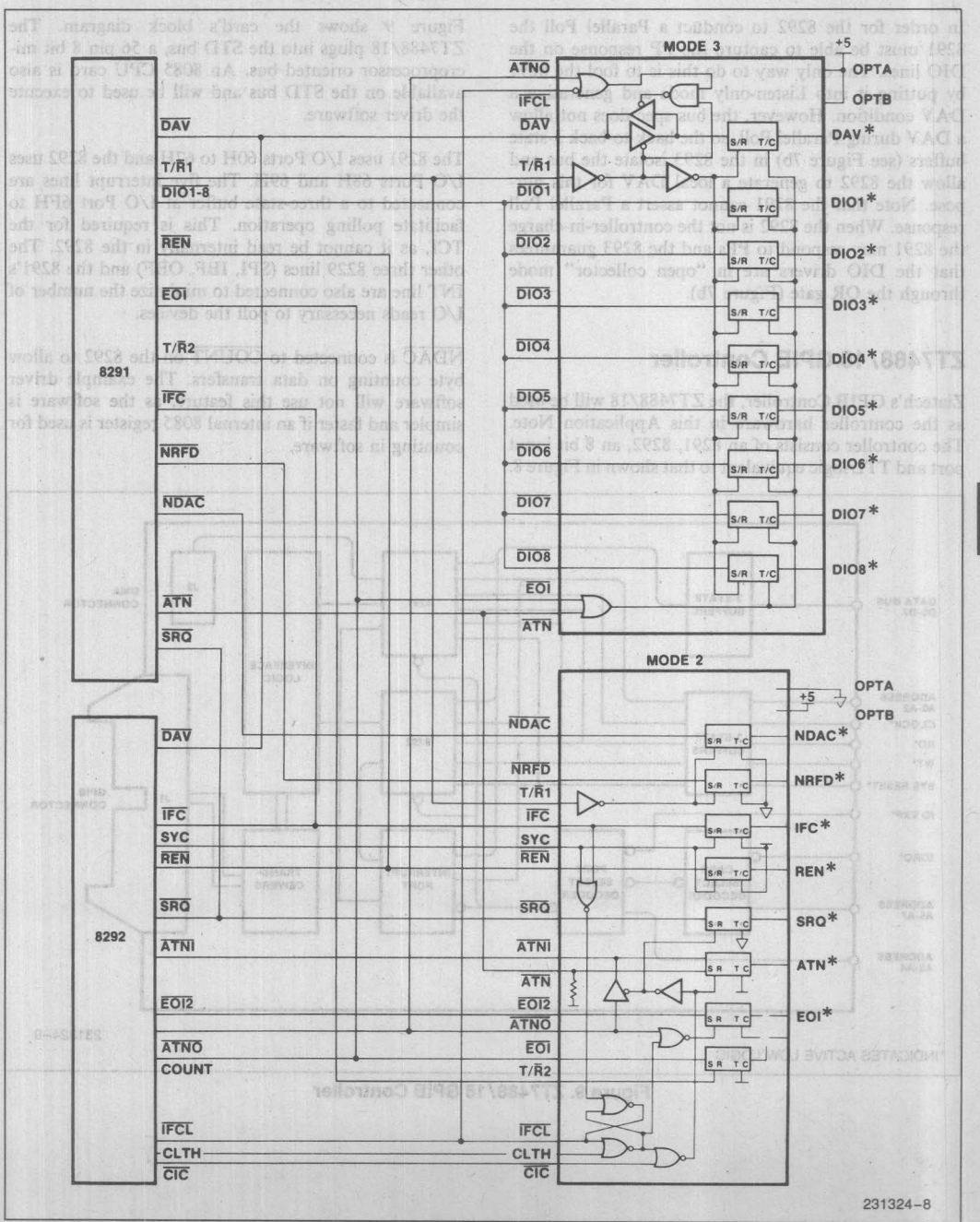


Figure 8. Talker/Listener/Controller

In order for the 8292 to conduct a Parallel Poll the 8291 must be able to capture the PP response on the DIO lines. The only way to do this is to fool the 8291 by putting it into Listen-only mode and generating a DAV condition. However, the bus spec does not allow a DAV during Parallel Poll, so the back-to-back 3-state buffers (see Figure 7b) in the 8293 isolate the bus and allow the 8292 to generate a local DAV for this purpose. Note that the 8291 cannot assert a Parallel Poll response. When the 8292 is not the controller-in-charge the 8291 may respond to PPs and the 8293 guarantees that the DIO drivers are in "open collector" mode through the OR gate (Figure 7b).

ZT7488/18 GPIB Controller

Ziatech's GPIB Controller, the ZT7488/18 will be used as the controller hardware in this Application Note. The controller consists of an 8291, 8292, an 8 bit input port and TTL logic equivalent to that shown in Figure 8. Figure 9 shows the card's block diagram. The ZT7488/18 plugs into the STD bus, a 56 pin 8 bit microprocessor oriented bus. An 8085 CPU card is also available on the STD bus and will be used to execute the driver software.

The 8291 uses I/O Ports 60H to 67H and the 8292 uses I/O Ports 68H and 69H. The five interrupt lines are connected to a three-state buffer at I/O Port 6FH to facilitate polling operation. This is required for the TCI, as it cannot be read internally in the 8292. The other three 8229 lines (SPI, IBF, OBF) and the 8291's INT line are also connected to minimize the number of I/O reads necessary to poll the devices.

$\overline{\text{NDAC}}$ is connected to $\overline{\text{COUNT}}$ on the 8292 to allow byte counting on data transfers. The example driver software will not use this feature, as the software is simpler and faster if an internal 8085 register is used for counting in software.

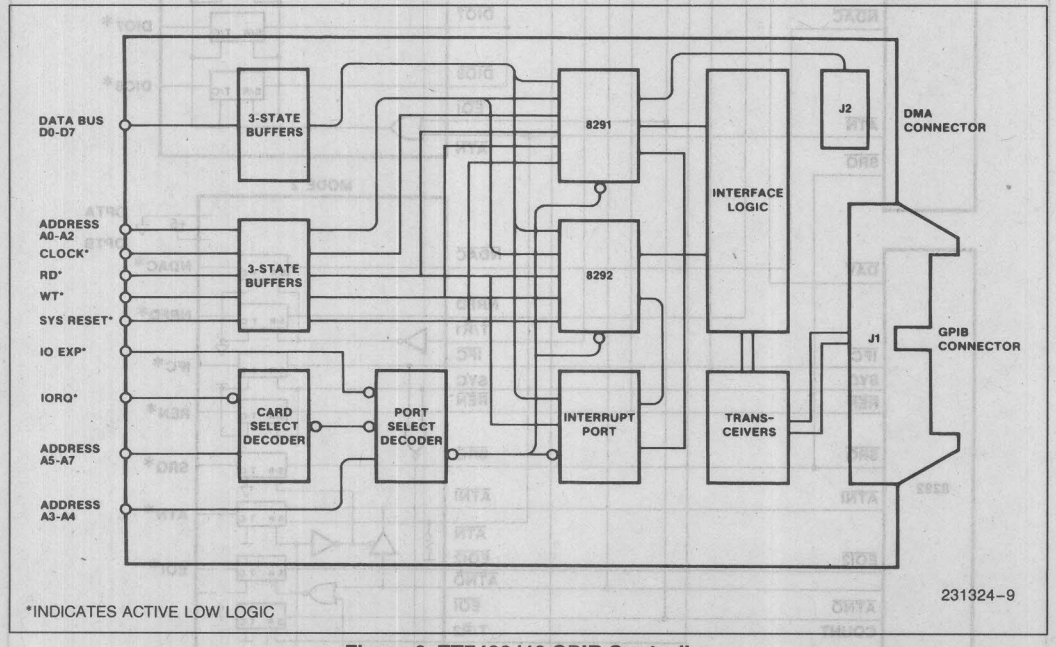


Figure 9. ZT7488/18 GPIB Controller

READ REGISTERS								PORT #	WRITE REGISTERS							
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	6ϕH	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
DATA IN									DATA OUT							
CPT	APT	GET	END	DEC	ERR	BO	BI	61H	CPT	APT	GET	END	DEC	ERR	BO	BI
INTERRUPT STATUS 1									INTERRUPT MASK 1							
INT	SPAS	LLO	REM	SPASC	LLOC	REMC	ADSC	62H	0	0	DMAO	DMAI	SPASC	LLOC	REMC	ADSC
INTERRUPT STATUS 2									INTERRUPT MASK 2							
S8	SRQS	S6	S5	S4	S3	S2	S1	63H	S8	rsv	S6	S5	S4	S3	S2	S1
SERIAL POLL STATUS									SERIAL POLL MODE							
ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN	64H	TO	LO	0	0	0	0	ADM1	ADM0
ADDRESS STATUS									ADDRESS MODE							
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	65H	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
COMMAND PASS THROUGH									AUX MODE							
X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	66H	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
ADDRESS 0									ADDRESS 0/1							
X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	67H	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
ADDRESS 1									EOS							

Figure 10. 8291 Registers

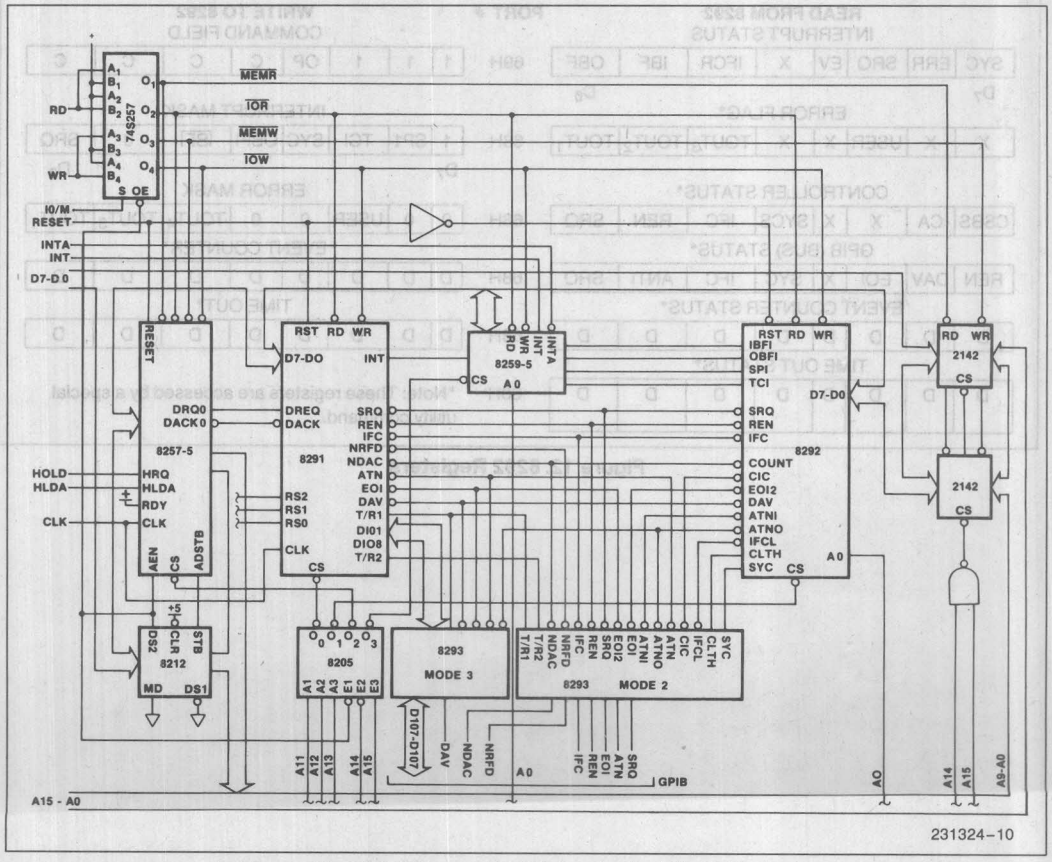


Figure 11. DMA/Interrupt GPIB Controller Block Diagram

The application example will not use DMA or interrupts; however, the Figure 11 block diagram includes these features for completeness.

The 8257-5 DMA chip can be used to transfer data between the RAM and the 8291 Talker/Listener. This mode allows a faster data rate on the GPIB and typically will depend on the 8291's EOS or EOI detection to terminate the transfer. The 8259-5 interrupt controller is used to vector the five possible interrupts for rapid software handling of the various conditions.

8292 COMMAND DESCRIPTION

This section discusses each command in detail and relates them to a particular GPIB activity. Recall that although the 8041A has only two read registers and one write register, through the magic of on-chip firmware the 8292 appears to have six read registers and five write registers. These are listed in Figure 12. Please see the 8292 data sheet for detailed definitions of each register. Note the two letter mnemonics to be used in later discussions. The CPU must not write into the 8292 while IBF (Input Buffer Full) is a one, as information will be lost.

Direct Commands

Both the Interrupt Mask (IM) and the Error Mask (EM) register may be directly written with the LSB of the address bus (A0) a "0". The firmware uses the MSB of the data written to differentiate between IM and EM.

LOAD INTERRUPT MASK

This command loads the Interrupt Mask with D7–D0. Note that D7 must be a "1" and that interrupts are enabled by a corresponding "1" bit in this register. IFC interrupt cannot be masked off; however, when the 8292 is the System Controller, sending an ABORT command will not cause an IFC interrupt.

READ FROM 8292								PORT #	WRITE TO 8292							
INTERRUPT STATUS									COMMAND FIELD							
SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF	69H	1	1	1	OP	C	C	C	C
D_7							D_0									
ERROR FLAG*									INTERRUPT MASK							
X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$	68H	1	SP1	TCI	SYC	OBFI	$\overline{IBFI}$	0	SRQ
									D_7							D_0
CONTROLLER STATUS*									ERROR MASK							
CSBS	CA	X	X	SYCS	IFC	REN	SRQ	68H	0	0	USER	0	0	$TOUT_4$	$TOUT_3$	$TOUT_1$
GPIB (BUS) STATUS*									EVENT COUNTER*							
REN	DAV	EOI	X	SYC	IFC	ANTI	SRQ	68H	D	D	D	D	D	D	D	D
EVENT COUNTER STATUS*									TIME OUT*							
D	D	D	D	D	D	D	D	68H	D	D	D	D	D	D	D	D
TIME OUT STATUS*																
D	D	D	D	D	D	D	D	68H	*Note: These registers are accessed by a special utility command.							

Figure 12. 8292 Registers

LOAD ERROR MASK

This command loads the Error Mask with D7–D0. Note that D7 must be a zero and that interrupts are enabled by a corresponding "1" bit in this register.

Utility Commands

These commands are used to read or write the 8292 registers that are not directly accessible. All utility commands are written with A0 = 1, D7 = D6 = D5 = 1, D4 = 0. D3–D0 specify the particular command. For writing into registers the general sequence is:

1) wait for IBF = 0 in Interrupt Status Register
2) write the appropriate command to the 8292,
3) write the desired register value to the 8292 with A0 = 1 with no other writes intervening,
4) wait for indication of completion from 8292 (IBF = 0).

For reading a register the general sequence is:

1) wait for IBF = 0 in Interrupt Status Register
2) write the appropriate command to the 8292
3) wait for a TCI (Task Complete Interrupt)
4) Read the value of the accessed register from the 8292 with A0 = 0.

WEVC—Write to Event Counter
(Command = 0E2H)

The byte written following this command will be loaded into the event counter register and event counter status for byte counting. The internal counter is incremented on a high to low transition of the COUNT (T1) input. In this application example $\overline{\text{NDAC}}$ is connected to count. The counter is an 8 bit register and therefore can count up to 256 bytes (writing 0 to the EC implies a count of 256). If longer blocks are desired, the main CPU must handle the interrupts every 256 counts and carefully observe the timing constraints.

Because the counter has a frequency range from 0 to 133 kHz when using a 6 MHz crystal, this feature may not be usable with all devices on the GPIB. The 8291 can easily transfer data at rates up to 250 kHz and even faster with some tuning of the system. There is also a 500 ns minimum high time requirement for COUNT which can potentially be violated by the 8291 in continuous acceptor handshake mode (i.e., TNDDV1 + TDVND2 − C = 350 + 350 = 700 max). When cable delays are taken into consideration, this problem will probably never occur.

When the 8292 has completed the command, IBF will become a "0" and will cause an interrupt if masked on.

WTOUT—Write to Time Out Register
(Command = 0E1H)

The byte written following this command will be used to determine the number of increments used for the time out functions. Because the register is 8 bits, the maximum time out is 256 time increments. This is probably enough for most instruments on the GPIB but is not enough for a manually stepped operation using a GPIB logic analyzer like Ziatech's ZT488. Also, the 488 Standard does not set a lower limit on how long a device may take to do each action. Therefore, any use of a time out must be able to be overridden (this is a good general design rule for service and debugging considerations).

The time out function is implemented in the 8292's firmware and will not be an accurate time. The counter counts backwards to zero from its initial value. The function may be enabled/disabled by a bit in the Error mask register. When the command is complete IBF will be set to a "0" and will cause an interrupt if masked on.

REVC—Read Event Counter Status
(Command = 0E3H)

This command transfers the content of the Event Counter to the DBBOUT register. The firmware then sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value from the 8292 with A0 = 0.

RINM—Read Interrupt Mask Register
(Command = 0E5H)

This command transfers the content of the Interrupt Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RERM—Read Error Mask Register
(Command = 0EAH)

This command transfers the content of the Error Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RCST—Read Controller Status Register
(Command = 0E6H)

This command transfers the content of the Controller Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

RTOUT—Read Time Out Status Register
(Command = 0E9H)

This command transfers the content of the Time Out Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

If this register is read while a time-out function is in process, the value will be the time remaining before time-out occurs. If it is read after a time-out, it will be zero. If it is read when no time-out is in process, it will be the last value reached when the previous timing occurred.

RBST—Read Bus Status Register
(Command = 0E7H)

This command causes the firmware to read the GPIB management lines, DAV and the SYC pin and place a copy in DBBOUT. TCI is set to "1" and will cause an interrupt if masked on. The CPU may read the value.

RERF—Read Error Flag Register
(Command = 0E4H)

This command transfers the content of the Error Flag register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

This register is also placed in DBBOUT by an IACK command if ERR remains set. TCI is set to "1" in this case also.

IACK—Interrupt Acknowledge
(Command = A1 A2 A3 A4 1 A5 1 1)

This command is used to acknowledge any combinations of the five SPI interrupts (A1–A5): SYC, ERR, SRQ, EV, and IFCR. Each bit A1–A5 is an individual acknowledgement to the corresponding bit in the Interrupt Status Register. The command clears SPI but it will be set again if all of the pending interrupts were not acknowledged.

If A2 (ERR) is "1", the Error Flag register is placed in DBBOUT and TCI is set. The CPU may then read the Error Flag without issuing an RERF command.

Operation Commands

The following diagram (Figure 13) is an attempt to show the interrelationships among the various 8292 Operation Commands. It is not meant to replace the complete controller state diagram in the IEEE Standard.

RST—Reset (Command = 0F2H)

This command has the same effect as an external reset applied to the chip's pin #4. The 8292's actions are:

1) All outputs go to their electrical high state. This means that SPI, TCI, OBFI, $\overline{\text{IBFI}}$, CLTH will be TRUE and all other GPIB signals will be FALSE.
2) The 8292's firmware will cause the above mentioned five signals to go FALSE after approximately 17.5 μs (at 6 MHz).
3) These registers will be cleared: Interrupt Status, Interrupt Mask, Error Mask, Time Out, Event Counter, Error Flag.
4) If the 8292 is the System Controller (SYC is TRUE), then IFC will be sent TRUE for approximately 100 μs and the Controller function will end up in charge of the bus. If the 8292 is not the System Controller then it will end up in an Idle state.
5) TCI will not be set.

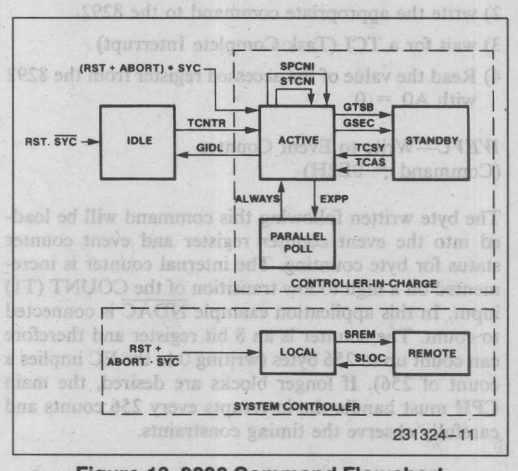


Figure 13. 8292 Command Flowchart

RSTI—Reset Interrupts (Command = 0F3)

This command clears all pending interrupts and error flags. The 8292 will stop waiting for actions to occur (e.g., waiting for ATN to go FALSE in a TCNTR command or waiting for the proper handshake state in a TCSY command). TCI will not be set.

ABORT—Abort all operations and Clear Interface
(Command = 0F9H)

If the 8292 is not the System Controller this command acts like a NOP and flags a USER ERROR in the Error Flag Register. No TCI will occur.

If the 8292 is the system Controller then IFC is set TRUE for approximately 100 μs and the 8292 becomes the Controller-in-Charge and asserts ATN. TCI will be set, only if the 8292 was NOT the CIC.

STCNI—Start Counter Interrupts
(Command = 0FEH)

Enables the EV Counter Interrupt. TCI will not be set. Note that the counter must be enabled by a GSEC command.

SPCNI—Stop Counter Interrupts
(Command = 0F0H)

The 8292 will not generate an EV interrupt when the counter reaches 0. Note that the counter will continue counting. TCI will not be set.

SREM—Set Interface to Remote Control
(Command = 0F8H)

If the 8292 is the System Controller, it will set REN and TCI TRUE. Otherwise it only sets the User Error Flag.

SLOC—Set Interface to Local Mode
(Command = 0F7H)

If the 8292 is the System Controller, it will set REN FALSE and TCI TRUE. Otherwise, it only sets the User Error Flag.

EXPP—Execute Parallel Poll
(Command = 0F5H)

If not Controller-in-Charge, the 8292 will treat this as a NOP and does not set TCI. If it is the Controller-in-Charge then it sets IDY (EOI & ATN) TRUE and generates a local DAV pulse (that never reaches the GPIB because of gates in the 8293). If the 8291 is configured as a listener, it will capture the Parallel Poll Response byte in its data register. TCI is not generated, the CPU must detect the BI (Byte In) from the 8291. The 8292 will be ready to accept another command before the BI occurs; therefore the 8291's BI serves as a task complete indication.

GTSB—Go To Standby (Command = 0F6H)

If the 8292 is not the Controller-in-Charge, it will treat this command as a NOP and does not set TCI TRUE. Otherwise, it goes to Controller Standby State (CSBS), sets ATN FALSE and TCI TRUE. This command is used as part of the Send, Receive, Transfer and Serial Poll System commands (see next section) to allow the addressed talker to send data/status.

If the data transfer does not start within the specified Time-Out, the 8292 sets TOUT2 TRUE in the Error Flag Register and sets SPI (if enabled). The controller continues waiting for a new command. The CPU must decide to wait longer or to regain control and take corrective action.

GSEC—Go To Standby and Enable Counting
(Command = 0F4H)

This command does the same things as GTSB but also initializes the event counter to the value previously stored in the Event Counter Register (default value is 256) and enables the counter. One may wire the count input to $\overline{\text{NDAC}}$ to count bytes. When the counter reaches zero, it sets EV (and SPI if enabled) in Interrupt Status and will set EV every 256 bytes thereafter. Note that there is a potential loss of count information if the CPU does not respond to the EV/SPI before another 256 bytes have been transferred. TCI will be set at the end of the command.

TCSY—Take Control Synchronously
(Command = 0FDH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it waits for the proper handshake state and sets ATN TRUE. The 8292 will set TOUT3 if the handshake never assumes the correct state and will remain in this command until the handshake is proper or a RSTI command is issued. If the 8292 successfully takes control, it sets TCI TRUE.

This is the normal way to regain control at the end of a Send, Receive, Transfer or Serial Poll System Command. If TCSY is not successful, then the controller must try TCAS (see warning below).

TCAS—Take Control Asynchronously
(Command = 0FCH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it arbitrarily sets ATN TRUE and ECI TRUE. Note that this action may cause devices on the bus to lose a data byte or cause them to interpret a data byte as a command byte. Both Actions can result in anomalous behavior. TCAS should be used only in emergencies. If TCAS fails, then the System Controller will have to issue an ABORT to clean things up.

GIDL—Go to Idle (Command = 0F1H)

If the 8292 is not the Controller in Charge and Active, then it treats this command as a NOP and does not set TCI. Otherwise, it sets ATN FALSE, becomes Not Controller in Charge, and sets TCI TRUE. This command is used as part of the Pass Control System Command.

TCNTR—Take (Receive) Control (Command = 0FAH)

If the 8292 is not Idle, then it treats this command as a NOP and does not set TCI. Otherwise, it waits for the current Controller-in-Charge to set ATN FALSE. If this does not occur within the specified Time Out, the 8292 sets TOUT1 in the Error Flag Register and sets SPI (if enabled). It will not proceed until ATN goes false or it receives an RSTI command. Note that the Controller in Charge must previously have sent this controller (via the 8291's command pass through register) a Pass Control message. When ATN goes FALSE, the 8292 sets CIC, ATN and TCI TRUE and becomes Active.

SOFTWARE DRIVER OUTLINE

The set of system commands discussed below is shown in Figure 14. These commands are implemented in software routines executed by the main CPU.

The following section assumes that the Controller is the System Controller and will not Pass Control. This is a valid assumption for 99+% of all controllers. It also assumes that no DMA or Interrupts will be used. SYC (System Control Input) should not be changed after Power-on in any system—it adds unnecessary complexity to the CPU's software.

In order to use polling with the 8292 one must enable TCI but not connect the pin to the CPU's interrupt pin. TCI must be readable by some means. In this application example it is connected to bit 1 port 6FH on the ZT7488/18. In addition, the other three 8292 interrupt lines and the 8291 interrupt are also on that port (SPI-Bit 2, $\overline{\text{IBFI}}$-Bit 4, OBFI-Bit 3, 8291 INT-Bit 0).

These drivers assume that only primary addresses will be used on the GPIB. To use secondary addresses, one must modify the test for valid talk/listen addresses (range macro) to include secondaries.

INIT	INITIALIZATION
Talker/Listener	
SEND	SEND DATA
RECV	RECEIVE DATA
XFER	TRANSFER DATA
Controller	
TRIG	GROUP EXECUTE TRIGGER
DCLR	DEVICE CLEAR
SPOL	SERIAL POLL
PPEN	PARALLEL POLL ENABLE
PPDS	PARALLEL POLL DISABLE
PPUN	PARALLEL POLL UNCONFIGURE
PPOL	PARALLEL POLL
PCTL	PASS CONTROL
RCTL	RECEIVE CONTROL
SRQD	SERVICE REQUESTED
System Controller	
REME	REMOTE ENABLE
LOCL	LOCAL
IFCL	ABORT/INTERFACE CLEAR

Figure 14. Software Drive Routines

Initialization

8292—Comes up in Controller Active State when SYC is TRUE. The only initialization needed is to enable the TCI interrupt mask. This is done by writing 0A0H to Port 68H.

8291—Disable both the major and minor addresses because the 8291 will never see the 8292's commands/addresses (refer to earlier hardware discussion). This is done by writing 60H and 0E0H to Port 66H.

Set Address Mode to Talk-only by writing 80H to Port 64H.

Set internal counter to 3 MHz to match the clock input coming from the 8085 by writing 23H to Port 65H. High speed mode for the handshakes will not be used here even though the hardware uses three-state drivers.

No interrupts will be enabled now. Each routine will enable the ones it needs for ease of polling operation. The INT bit may be read through Port 6FH. Clear both interrupt mask registers.

Release the chip's initialization state by writing 0 to Port 65H.

```
INIT:
 Enable-8292                    ;Set up In. pins for Port 6FH
  Enable TCI                    ;Task complete must be on
 Enable-8291
  Disable major address         ;In controller usage, the 8291
  Disable minor address         ;Is set to talk only and/or listen only
  ton                           ;Talk only is our rest state
  Clock frequency               ;3 MHz in this ap note example
  All interrupts off
  Immediate execute pon         ;Releases 8291 from init. state
```

Talker/Listener Routines

SEND DATA

SEND <listener list pointer> <count> <EOS> <data buffer pointer>

This system command sends data from the CPU to one or more devices. The data is usually a string of ASCII characters, but may be binary or other forms as well. The data is device-specific.

My Talk Address (MTA) must be output to satisfy the GPIB requirement of only one talker at a time (any other talker will stop when MTA goes out). The MTA is not needed as far as the 8291 is concerned—it will be put into talk-only mode (ton).

This routine assumes a non-null listener list in that it always sends Univeral Unlisten. If it is desired to send data to the listeners previously addressed, one could add a check for a null list and not send UNL. Count must be 255 or less due to an 8 bit register. This routine also always uses an EOS character to terminate the string output; this could easily be eliminated and rely on the count. Items in brackets ( ) are optional and will not be included in the actual code in Appendix A.

```
SEND:
 Output-to-8291 MTA, UNL          ;We will talk, nobody listen
 Put EOS into 8291                ;End of string compare character
 While 20H ≤ listener ≤ 3EH       ;GPIB listen addresses are
  output-to-8291 listener         ;"space" thru ">" ASCII
  Increment listen list pointer   ;Address all listeners
 Output-to-8292 GTSB              ;8292 stops asserting ATN, go to standby
 Enable-8291
  Output EOI on EOS sent          ;Send EOI along with EOS character
 If count < > 0 then
  While not (end or count = 0)    ;Wait for EOS or end of count
  (could check tout 2 here)       ;Optionally check for stuck bus-tout 2
   Output-to-8291 data            ;Output all data, one byte at a time
   Increment data buffer pointer  ;8085 CREG will count for us
   Decrement count
 Output-to-8292 TCSY              ;8292 asserts ATN, take control sync.
 (If tout3 then take control async) ;If unable to take control sync.
 Enable 8291                      ;Restore 8291 to standard condition
  No output EOI on EOS sent
 Return
```

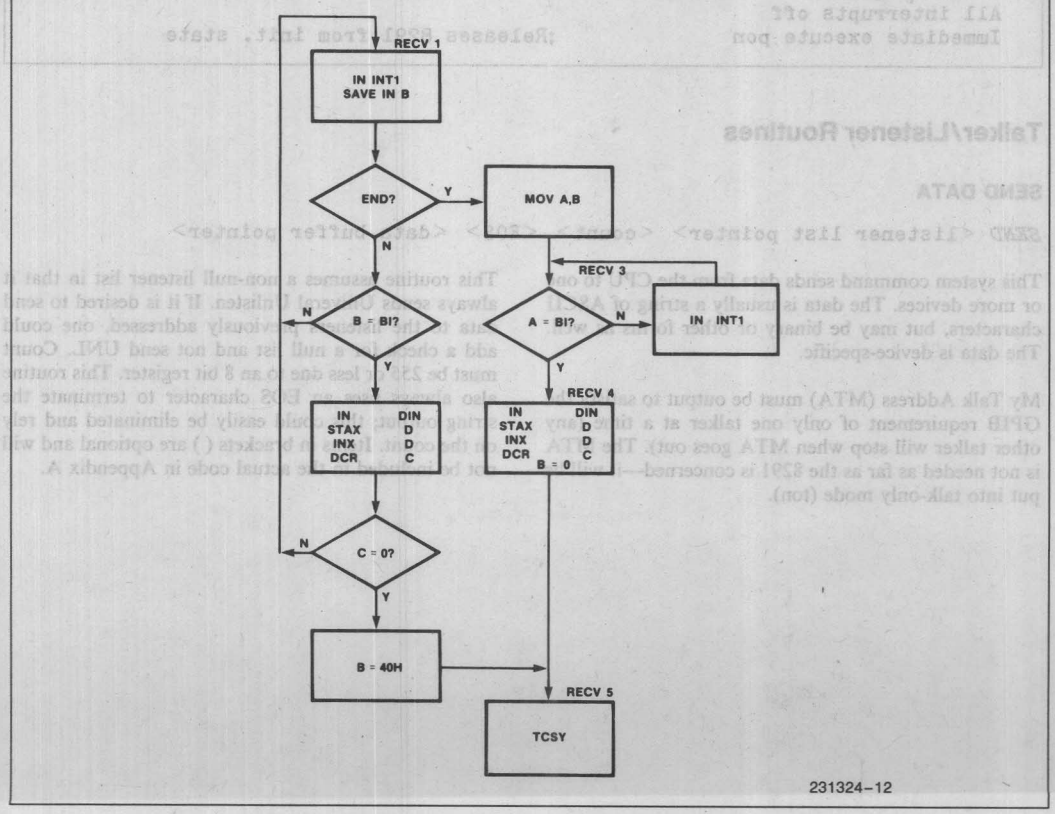


Figure 15. Flowchart for Receive Ending Conditions

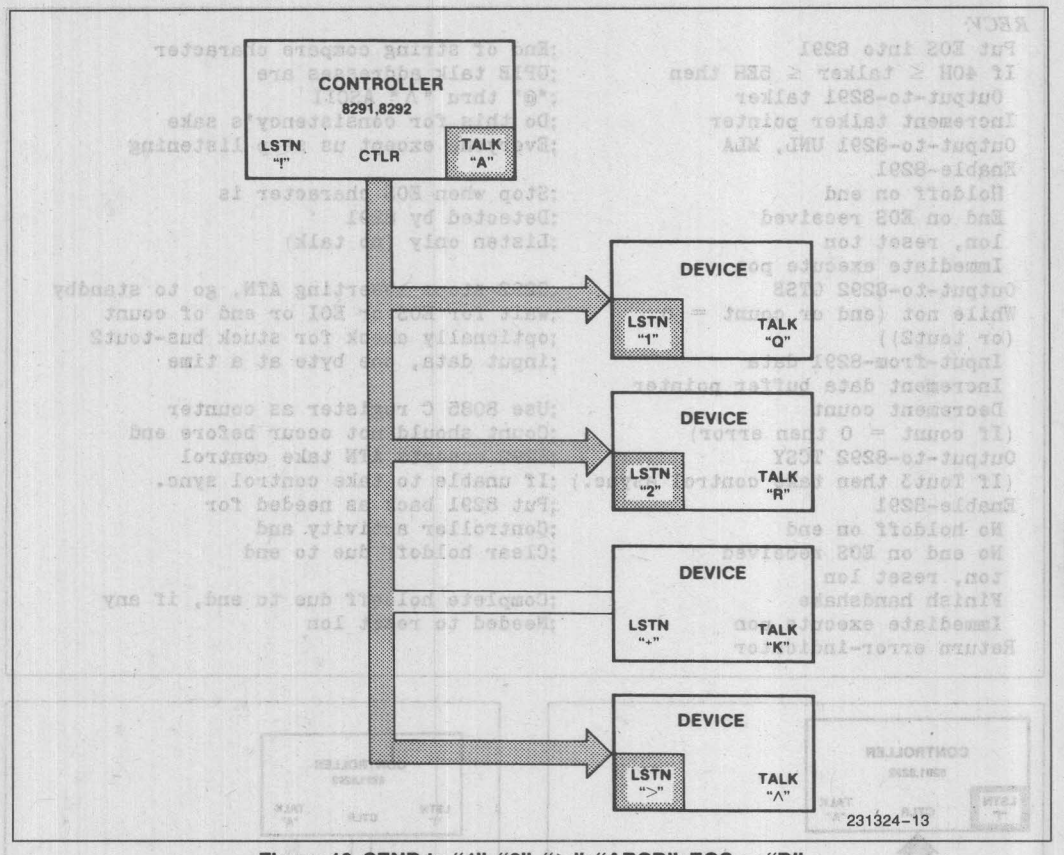


Figure 16. SEND to "1", "2", ">"; "ABCD"; EOS = "D"

RECEIVE DATA

RECV <talker> <count> <EOS> <data buffer pointer>

This system command is used to input data from a device. The data is typically a string of ASCII characters.

This routine is the dual of SEND. It assumes a new talker will be specified, a count of less than 257, and an EOS character to terminate the input. EOI received will also terminate the input. Figure 15 shows the flow chart for the RECV ending conditions. My Listen Address (MLA) is sent to keep the GPIB transactions totally regular to facilitate analysis by a GPIB logic analyzer like the Ziatech ZT488. Otherwise, the bus would appear to have no listener even though the 8291 will be listening.

Note that although the count may go to zero before the transmission ends, the talker will probably be left in a strange state and may have to be cleared by the controller. The count ending of RECV is therefore used as an error condition in most situations.

```
RECV:
 Put EOS into 8291                          ;End of string compare character
 If 40H ≤ talker ≤ 5EH then                 ;GPIB talk addresses are
  Output-to-8291 talker                     ;"@" thru "^" ASCII
 Increment talker pointer                   ;Do this for consistency's sake
 Output-to-8291 UNL, MLA                    ;Everyone except us stop listening
 Enable-8291
  Holdoff on end                            ;Stop when EOS character is
  End on EOS received                       ;Detected by 8291
  lon, reset ton                            ;Listen only (no talk)
  Immediate execute pon
 Output-to-8292 GTSB                        ;8292 stops asserting ATN, go to standby
 While not (end or count = 0                ;wait for EOS or EOI or end of count
 (or tout2))                                ;optionally check for stuck bus-tout2
  Input-from-8291 data                      ;input data, one byte at a time
  Increment data buffer pointer
  Decrement count                           ;Use 8085 C register as counter
 (If count = 0 then error)                  ;Count should not occur before end
 Output-to-8292 TCSY                        ;8292 asserts ATN take control
 (If Tout3 then take control async.)        ;If unable to take control sync.
 Enable-8291                                ;Put 8291 back as needed for
  No holdoff on end                         ;Controller activity and
  No end on EOS received                    ;Clear holdoff due to end
  ton, reset lon
  Finish handshake                          ;Complete holdoff due to end, if any
  Immediate execute pon                     ;Needed to reset lon
 Return error-indicator
```

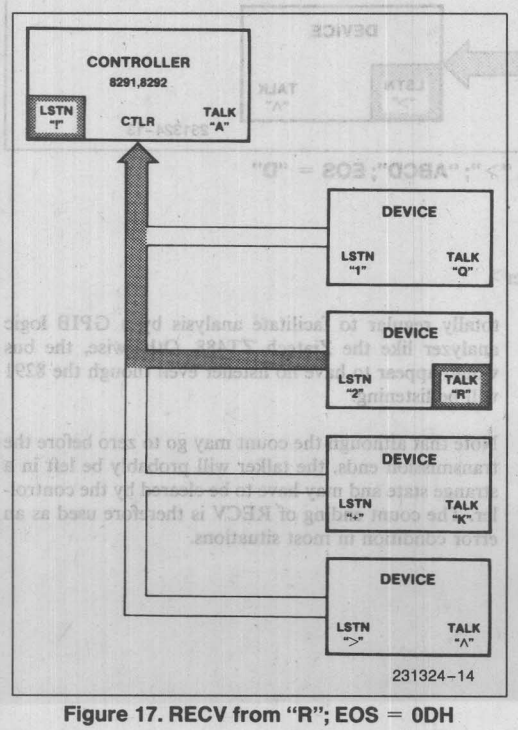


Figure 17. RECV from "R"; EOS = 0DH

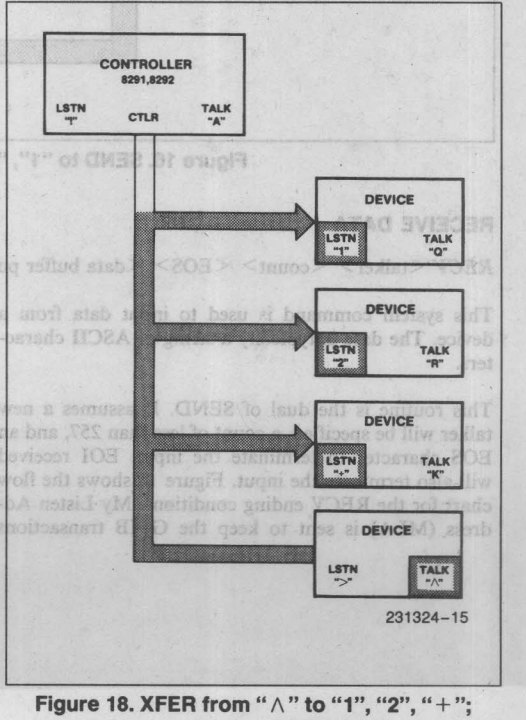


Figure 18. XFER from "^" to "1", "2", "+"; EOS = 0DH

TRANSFER DATA

XFER <Talker> <Listener list> <EOS>

This system command is used to transfer data from a talker to one or more listeners where the controller does not participate in the transfer of the ASCII data. This is accomplished through the use of the 8291's continuous acceptor handshake mode while in listen-only.

This routine assumes a device list that has the ASCII talker address as the first byte and the string (one or more) or ASCII listener addresses following. The EOS character or an EOI will cause the controller to take take control synchronously and thereby terminate the transfer.

```
XFER:
 Output-to-8291: Talker, UNL          ;Send talk address and unlisten
 While 20H ≤ listen ≤ 3EH
  Output-to-8291: Listener            ;Send listen address
  Increment listen list pointer
 Enable-8291
  lon, no ton                         ;Controller is pseudo listener
  Continuous AH mode                  ;Handshake but don't capture data
  End on EOS received                 ;Capture EOS as well as EOI
  Immediate execute PON               ;Initialize the 8291
 Put EOS into 8291                    ;Set up EOS character
 Output-to-8292: GTSB                 ;Go to standby
                                      ;8292 waits for EOS or EOI and then
 Upon end (or tout2) then
  Take control synchronously          ;Regains control
 Enable-8291                          ;Go to Ready for Data
  Finish handshake
  Not continuous AH mode
  Not END on EOS received
  ton
  Immediate execute pon
 Return
```

Controller

GROUP EXECUTE TRIGGER

TRIG <Listener list>

This system command causes a group execute trigger (GET) to be sent to all devices on the listener list. The intended use is to synchronize a number of instruments.

```
TRIG:
 Output-to-8291 UNL                   ;Everybody stop listening
 While 20H ≤ listener ≤ 3EH           ;Check for valid listen address
  Output-to-8291 Listener             ;Address each listener
  Increment listen list pointer       ;Terminate on any non-valid character
 Output-to-8291 GET                   ;Issue group execute trigger
 Return
```

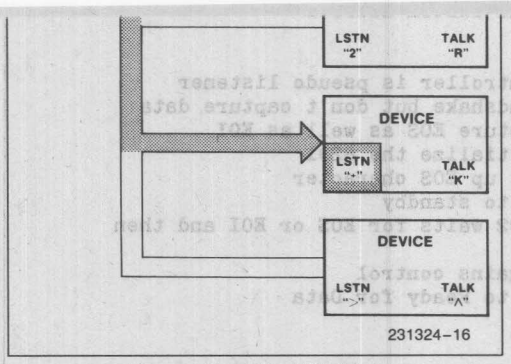


Figure 19. TRIG "1", "+"

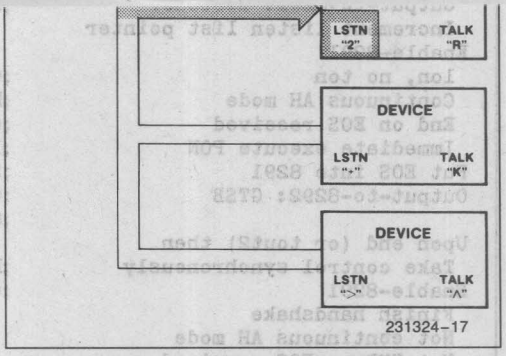


Figure 20. DCLR "1", "2"

DEVICE CLEAR

DCLR <Listener list>

This system command causes a device clear (SDC) to be sent to all devices on the listener list. Note that this is not intended to clear the GPIB interface of the device, but should clear the device-specific logic.

```
DCLR:
 Output-to-8291 UNL                    ;Everybody stop listening
 While 20H ≤ listener ≤ 3EH            ;Check for valid listen address
  Output-to-8291 Listener              ;Address each listener
  Increment listen list pointer        ;Terminate on any non-valid character
 Output-to-8291 SDC                    ;Selective device clear
 Return
```

SERIAL POLL

SPOL <Talker list> <status buffer pointer>

This system command sequentially addresses the designated devices and receives one byte of status from each. The bytes are stored in the buffer in the same order as the devices appear on the talker list. MLA is output for completeness.

```
SPOL:
 Output-to-8291 UNL, MLA, SPE        ;Unlisten, we listen, serial poll enable
                                      ;Only one byte of serial poll
                                      ;Status wanted from each talker
 While 40H ≤ talker ≤ 5 EH            ;Check for valid transfer
  Output-to-8291 talker               ;Address each device to talk
  Increment talker list pointer       ;One at a time
  Enable-8291
   lon, reset ton                     ;Listen only to get status
   Immediate execute pon              ;This resets ton
  Output-to-8292 GTSB                 ;Go to standby
  Wait for data in (BI)               ;Serial poll status byte into 8291
  Output-to-8292 TCSY                 ;Take control synchronously
  Input-from-8291 data                ;Actually get data from 8291
  Increment buffer pointer
  Enable 8291
   ton, reset lon
   Immediate execute pon              ;Reset lon
  Output-to-8291 SPD                  ;Send serial poll disable after all
                                       devices polled
  Return
```

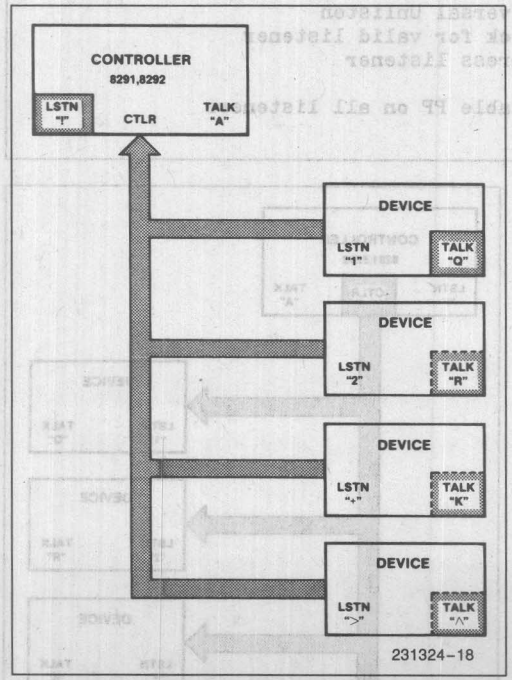


Figure 21. SPOL "Q", "R", "K", "∧"

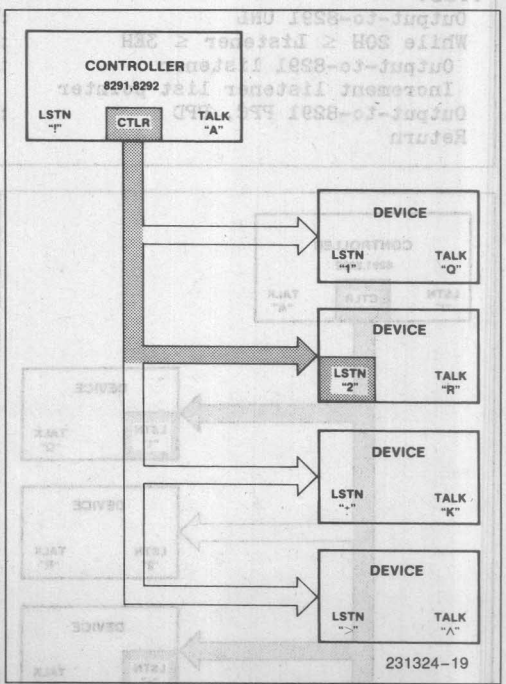


Figure 22. PPEN "2"; $iP_3P_2P_1 = 0111B$

PARALLEL POLL ENABLE

PPEN<Listener list> <Configuration Buffer pointer>

This system command configures one or more devices to respond to Parallel Poll, assuming they implement subset PP1. The configuration information is stored in a buffer with one byte per device in the same order as

devices appear on the listener list. The configuration byte has the format XXXXIP3P2P1 as defined by the IEEE Std. P3P2P1 indicates the bit # to be used for a response and I indicates the assertion value. See Sec. 2.9.3.3 of the Std. for more details.

```
PPEN:
 Output-to-8291 UNL                    ;Universal unlisten
 While 20H ≤ Listener ≤ 3EH            ;Check for valid listener
  Output-to-8291 listener              ;Stop old listener, address new
  Output-to-8291 PPC, (PPE or data)    ;Send parallel poll info
  Increment listener list pointer      ;Point to next listener
  Increment buffer pointer             ;One configuration byte per listener
 Return
```

PARALLEL POLL DISABLE

PPDS<listener list>

This system command disables one or more devices from responding to a Parallel Poll by issuing a Parallel Poll Disable (PPD). It does not deconfigure the devices.

```
PPDS:
 Output-to-8291 UNL                    ;Universal Unlisten
 While 20H ≤ Listener ≤ 3EH            ;Check for valid listener
  Output-to-8291 listener              ;Address listener
  Increment listener list pointer
 Output-to-8291 PPC, PPD               ;Disable PP on all listeners
 Return
```

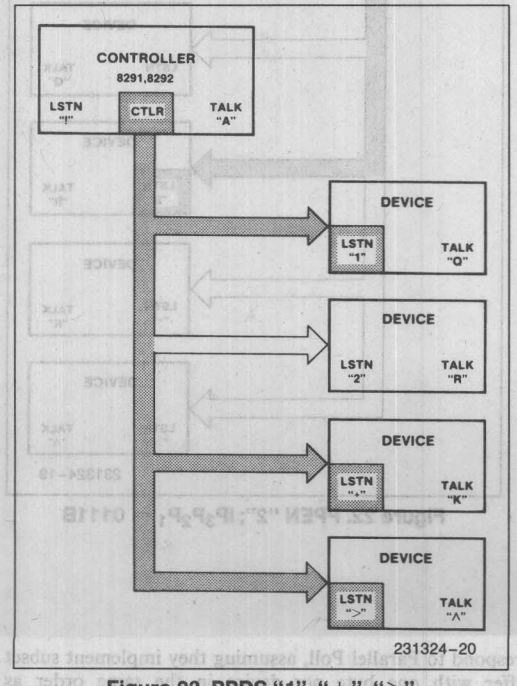


Figure 23. PPDS "1", "+", ">"

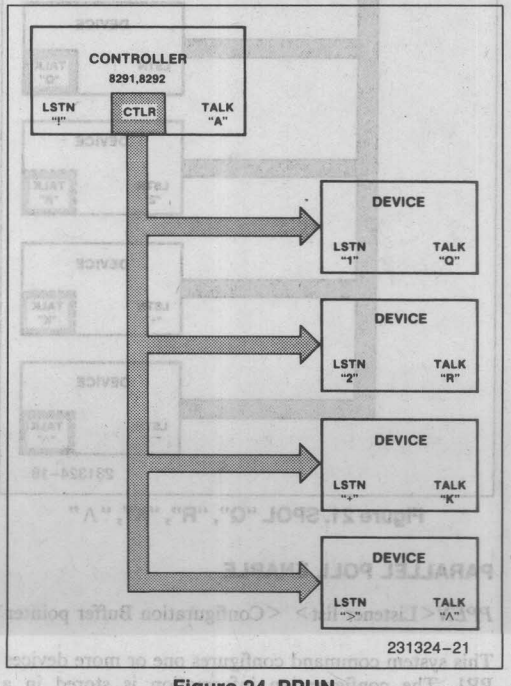


Figure 24. PPUN

PARALLEL POLL UNCONFIGURE

PPUN

This system command deconfigures the Parallel Poll response of all devices by issuing a Parallel Poll Unconfigure message.

```
PPUN:
 Output-to-8291 PPU                ;Unconfigure all parallel poll
 Return
```

CONDUCT A PARALLEL POLL

PPOL

This system command causes the controller to conduct a Parallel Poll on the GPIB for approximately 12.5 μsec (at 6 MHz). Note that a parallel poll does not use the handshake; therefore, to ensure that the device knows whether or not its positive response was observed by the controller, the CPU should explicitly acknowledge each device by a device-dependent data string. Otherwise, the response bit will still be set when the next Parallel Poll occurs. This command returns one byte of status.

```
PPOL:
 Enable-8291
  lon                              ;Listen only
  Immediate execute pon            ;This resets ton
 Output-to-8292 EXPP               ;Execute parallel poll
 Upon BI                           ;When byte is input
  Input-from-8291 data             ;Read it
 Enable-8291
  ton                              ;Talk only
  Immediate execute pon            ;This resets lon
 Return Data (status byte)
```

PASS CONTROL

PCTL<talker>

This system command allows the controller to relinquish active control of the GPIB to another controller. Normally some software protocol should already have informed the controller to expect this, and under what conditions to return control. The 8291 must be set up to become a normal device and the CPU must handle all commands passed through, otherwise control cannot be returned (see Receive Control below). The controller will go idle.

```
PCTL:
 If 40H ≤ talker ≤ 5EH then
  if talker < > MTA then           ;Cannot pass control to myself
   output-to-8291 talker, TCT      ;Take control message to talker
   Enable-8291                     ;Set up 8291 as normal device
    not ton, not lon
    Immediate execute pon          ;Reset ton and lon
    My device address, mode 1      ;Put device number in Register 6
    Undefined command pass through ;Required to receive control
    (Parallel Poll Configuration)  ;Optional use of PP
  Output-to-8292 GIDL              ;Put controller in idle
 Return
```

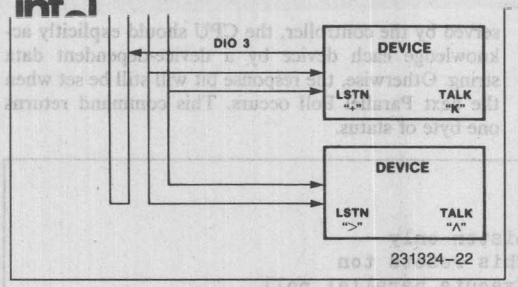


Figure 25. PPOL

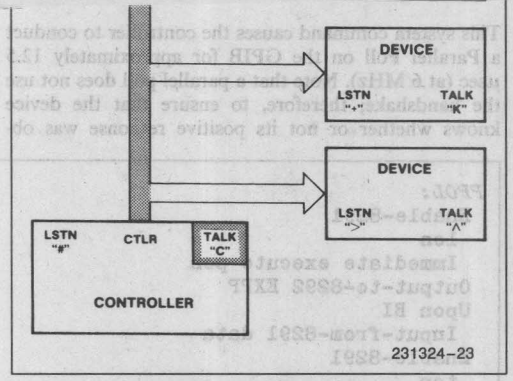


Figure 26. PCTL "C"

RECEIVE CONTROL

RCTL

This system command is used to get control back from the current controller-in-charge if it has passed control to this inactive controller. Most GPIB systems do not use more than one controller and therefore would not need this routine.

To make passing and receiving control a manageable event, the system designer should specify a protocol whereby the controller-in-charge sends a data message to the soon-to-be-active controller. This message should give the current state of the system, why control is being passed, what to do, and when to pass control back. Most of these issues are beyond the scope of this Ap Note.

```
RCTL:
 Upon CPT                              ;Wait for command pass through bit in 8291
  If (command=TCT) then                ;If command is take control and
   If TA then                          ;We are talker addressed
    Enable-8291
     Disable major device number       ;Controller will use ton and lon
     ton                               ;Talk only mode
     Mask off interrupts
     Immediate execute pon
    Output-to-8292 TCNTR               ;Take (receive) control
    Enable-8291
     Valid command                     ;Release handshake
    Return valid
   Else
    Enable-8291
     Invalid command                   ;Not talker addr. so TCT not for us
  Else
   Enable-8291
    Invalid command                    ;Not TCT, so we don't care
 Return invalid
```

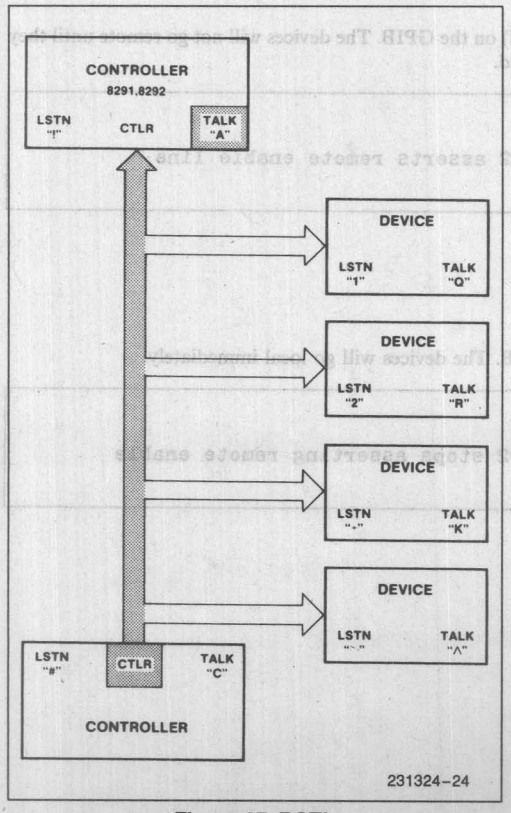


Figure 27. RCTL

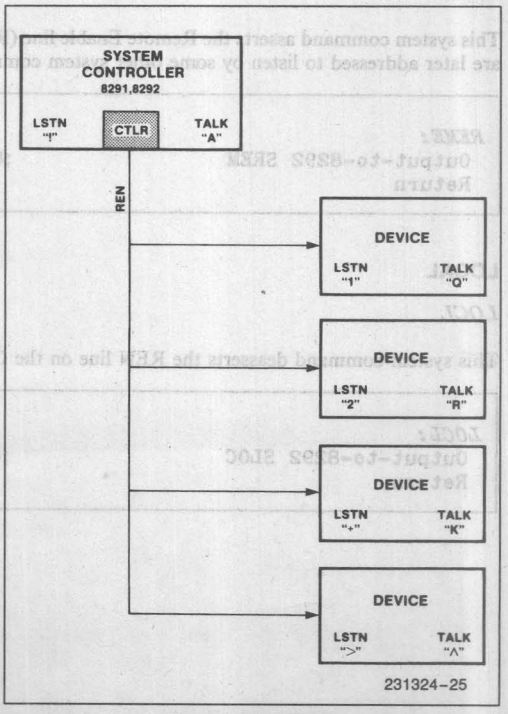


Figure 28. REME

SERVICE REQUEST

SRQD

This system command is used to detect the occurrence of a Service Request on the GPIB. One or more devices may assert SRQ simultaneously, and the CPU would normally conduct a Serial Poll after calling this routine to determine which devices are SRQing.

```
SRQD:
 If SRQ then                          ;Test 92 status bit
  Output-to-8292 IACK.SRQ             ;Acknowledge it
  Return SRQ
  Else return no SRQ
```

System Controller

REMOTE ENABLE

REME

This system command asserts the Remote Enable line (REN) on the GPIB. The devices will not go remote until they are later addressed to listen by some other system command.

```
REME:
 Output-to-8292 SREM                  ;8292 asserts remote enable line
 Return
```

LOCAL

LOCL

This system command deasserts the REN line on the GPIB. The devices will go local immediately.

```
LOCL:
 Output-to-8292 SLOC                  ;8292 stops asserting remote enable
 Return
```

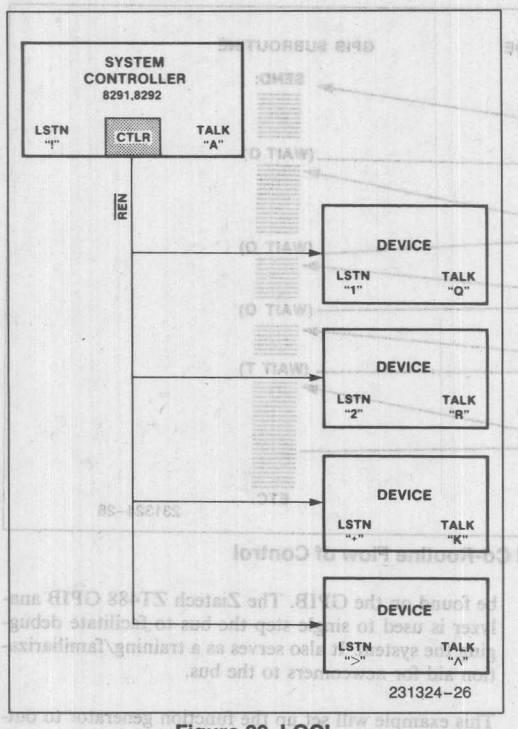


Figure 29. LOCL

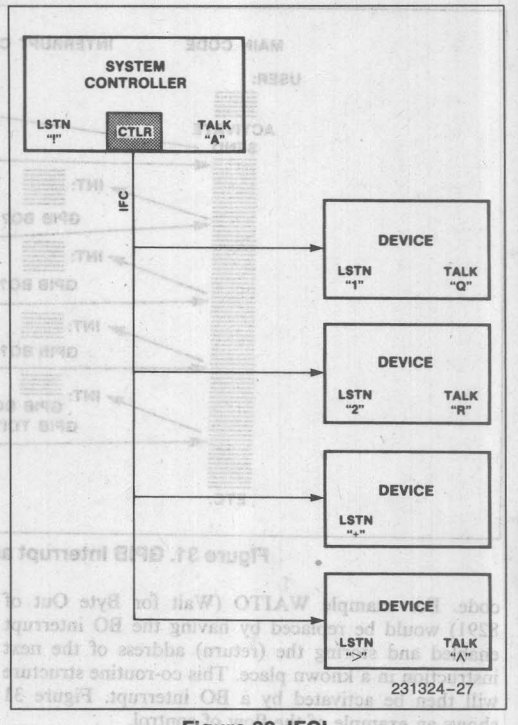


Figure 30. IFCL

INTERFACE CLEAR/ABORT

IFCL

This system command asserts the GPIB's Interface Clear (IFC) line for at least 100 microseconds. This causes all interface logic in all devices to go to a known state. Note that the device itself may or may not be reset, too. Most instruments do totally reset upon IFC. Some devices may require a DCLR as well as an IFCL to be completely reset. The (system) controller becomes Controller-in-Charge.

```
IFCL:
 Output-to-8292 ABORT              ;8292 asserts Interface Clear
 Return                            ;For 100 microseconds
```

INTERRUPTS AND DMA CONSIDERATIONS

The previous sections have discussed in detail how to use the 8291, 8292, 8293 chip set as a GPIB controller with the software operating in a polling mode and using programmed transfer of the data. This is the simplest mode of use, but it ties up the microprocessor for the duration of a GPIB transaction. If system design constraints do not allow this, then either Interrupts and/or DMA may be used to free up processor cycles.

The 8291 and 8292 provide sufficient interrupts that one may return to do other work while waiting for such things as 8292 Task Completion, 8291 Next Byte In, 8291 Last Byte Out, 8292 Service Request In, etc. The only difficulty lies in integrating these various interrupt sources and their matching routines into the overall system's interrupt structure. This is highly situation-specific and is beyond the scope of this Ap Note.

The strategy to follow is to replace each of the WAIT routines (see Appendix A) with a return to the main code and provide for the corresponding interrupt to bring the control back to the next section of GPIB

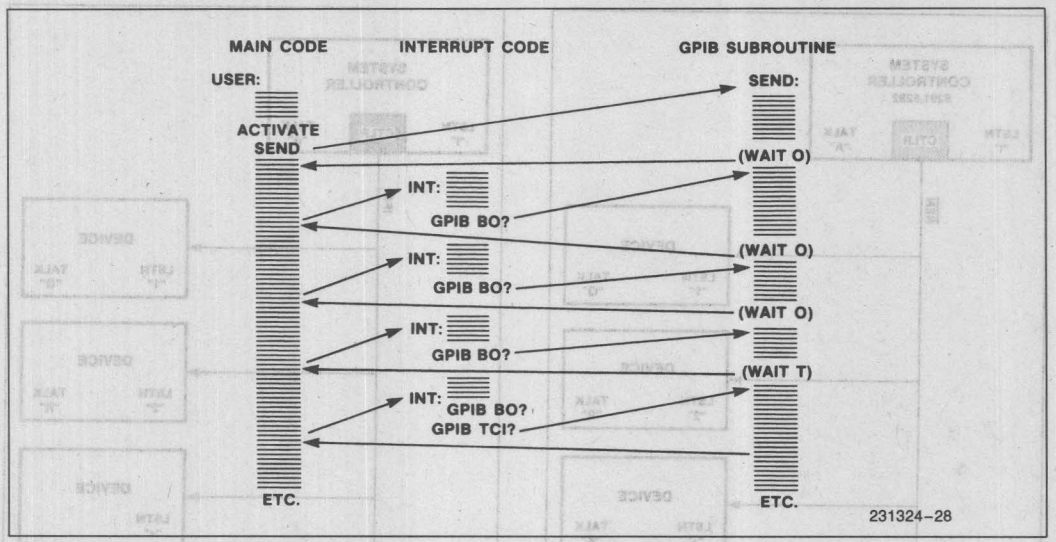


Figure 31. GPIB Interrupt and Co-Routine Flow of Control

code. For example WAITO (Wait for Byte Out of 8291) would be replaced by having the BO interrupt enabled and storing the (return) address of the next instruction in a known place. This co-routine structure will then be activated by a BO interrupt. Figure 31 shows an example of the flow of control.

DMA is also useful in relieving the processor if the average length of a data buffer is long enough to overcome the extra time used to set up a DMA chip. This decision will also be a function of a data rate of the instrument. The best strategy is to use the DMA to handle only the data buffer transfers on SEND and RECV and to do all the addressing and control just as shown in the driver descriptions.

Another major reason for using a DMA chip is to increase the data rate and therefore increase the overall transaction rate. In this case the limiting factor becomes the time used to do the addressing and control of the GPIB using software like that in Appendix A. The data transmission time becomes insignificant at DMA speeds unless extremely long buffers are used.

Refer to Figure 11 for a typical DMA and interrupt based design using the 8291, 8292, 8293. A system like this can achieve a 250K byte transfer rate while under DMA control.

APPLICATION EXAMPLE

This section will present the code required to operate a typical GPIB instrument set up as shown in Figure 32. The HP5328A universal counter and the HP3325 function generator are typical of many GPIB devices; however, there are a wide variety of software protocols to be found on the GPIB. The Ziatech ZT488 GPIB analyzer is used to single step the bus to facilitate debugging the system. It also serves as a training/familiarization aid for newcomers to the bus.

This example will set up the function generator to output a specific waveform, frequency and amplitude. It will then tell the counter to measure the frequency and Request Service (SRQ) when complete. The program will then read in the data. The assembled source code will be found at the end of Appendix A.

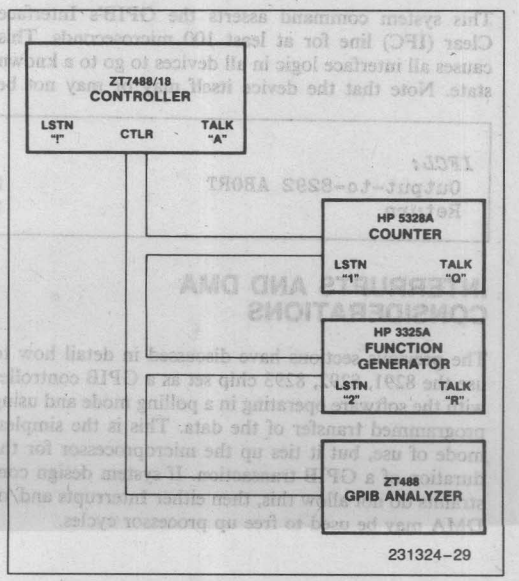


Figure 32. GPIB Example Configuration

```
SEND
 LSTN: "2", COUNT: 15, EOS: ODH, DATA: "FU1FR37KHAM2VO (CR)"
 ;SETS UP FUNCTION GEN. TO 37 KHZ SINE, 2 VOLTS PP
 ;COUNT EQUAL TO # CHAR IN BUFFER
 ;EOS CHARACTER IS (CR) = ODH = CARRIAGE

SEND
 LSTN: "1", COUNT: 6, EOS: "T" DATA: "PR4G7T"
 ;SETS UP COUNTER FOR P:INITIALIZE, F4: FREQ CHAN A
 ;         G7:0.1 HZ RESOLUTION, T:TRIGGER AND SRQ
 ;COUNT IS EQUAL TO # CHAR

WAIT FOR SRQ
SPOL TALK: "Q", DATA: STATUS 1
 ;CLEARS THE SRO_IN THIS EXAMPLE ONLY FREQ CTR ASSERTS SRQ

RECV TALK: "Q", COUNT: 17, EOS: OAH,
 DATA: "+   37000.0E+0" (CR) (LF)
 ;GETS 17 BYTES OF DATA FROM COUNTER
 ;COUNT IS EXACT BUFFER LENGTH
 ;DATA SHOWN IS TPYICAL HP5328A READING THAT WOULD BE RECEIVED
```

CONCLUSION

This Application Note has shown a structured way to view the IEEE 488 bus and has given typical code sequences to make the Intel 8291, 8292, and 8293's behave as a controller of the GPIB. There are other ways to use the chip set, but whatever solution is chosen, it must be integrated into the overall system software.

The ultimate reference for GPIB questions is the IEEE Std 488 -1978 which is available from IEEE, 345 East 47th St., New York, NY, 10017. The ultimate reference for the 8292 is the source listing for it (remember it's a pre-programmed UPI-41A) which is available from INSITE, Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051.

3

APPENDIX A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0
GPIB CONTROLLER SUBROUTINES

LOC  OBJ          LINE        SOURCE STATEMENT

                     1 $TITLE('GPIB CONTROLLER SUBROUTINES')
                     2 ;
                     3 ; GPIB CONTROLLER SUBROUTINES
                     4
                     5 ;
                     6 ;       for Intel 8291, 8292 on ZT 7488/18
                     7 ;       Bert Forbes, Ziatech Corporation
                     8 ;       2410 Broad Street
                     9 ;       San Luis Obispo, CA, USA 93401
                    10 ;
                    11 ;
                    12 ;       General Definitions & Equates
                    13 ;       8291 Control Values
                    14 ;
1000                15          ORG     1000H   ; For ZT7488/18 w/8085
                    16 ;
0060                17 PRT91    EQU     60H     ;8291 Base Port #
                    18 ;
                    19 ;       Reg #0 Data in & Data out
0060                20 DIN      EQU     PRT91+0 ;91 Data in reg
0060                21 DOUT     EQU     PRT91+0 ;91 Data out reg
                    22 ;
                    23 ;       Reg # 1 Interrupt 1 Constants
0061                24 INT1     EQU     PRT91+1 ;INT Reg 1
0061                25 INTM1    EQU     PRT91+1 ;INT Mask Reg. 1
0002                26 BOM      EQU     02      ;91 BO INTRP Mask
0001                27 BIM      EQU     01      ;91 BI INTRP Mask
0010                28 ENDMK    EQU     10H     ;91 END INTRP Mask
0080                29 CPT      EQU     80H     ;91 command pass thru int bit
                    30 ;
                    31 ;       Reg #2 Interrupt 2
0062                32 INT2     EQU     PRT91+2
                    33 ;
                    34 ;       Reg #4 Address Mode Constants
0064                35 ADRMD    EQU     PRT91+4 ;91 address mode register #
0080                36 TON      EQU     80H     ;91 talk only mode & not listen only
0040                37 LON      EQU     40H     ;91 listen only & not ton
00C0                38 TLON     EQU     0C0H    ;91 talk & listen only
0001                39 MODE1    EQU     01      ;mode 1 addressing for device
                    40
                    41 ;       Reg #4          (Read)  Address Status Register
0064                42 ADRST    EQU     PRT91+4 ;reg #4
0020                43 EOIST    EQU     20H
0002                44 TA       EQU     2
0001                45 LA       EQU     1       ;listener active
                    46 ;
                    47 ;       Reg #5  (Write) Auxillary Mode Register
0065                48 AUXMD    EQU     PRT91+5 ;91 auxillary mode register #
0023                49 CLKRT    EQU     23H     ;91 3 Mhz clock input
```

231324-30

```
0003                50 FNHSK   EQU     03      ;91 fininsh handshake command
0006                51 SDEOI   EQU     06      ;91 send EOI with next byte
0080                52 AXRA    EQU     80H     ;91 aux. reg A pattern
0001                53 HOHSK   EQU      1      ;91 hold off handshake on all bytes
0002                54 HOEND   EQU      2      ;91 hold off handshake on end
0003                55 CAHCY   EQU      3      ;91 continuous AH cycling
0004                56 EDEOS   EQU      4      ;91 end on EOS received
0008                57 EOIS    EQU      8      ;91 output EOI on EOS sent
000F                58 VSCMD   EQU     0FH     ;91 valid command pass through
0007                59 NVCMD   EQU     07H     ;91 invalid command pass through
00A0                60 AXRB    EQU     0A0H    ;Aux. reg. B pattern
0001                61 CPTEN   EQU     01H     ;command pass thru enable
                    62 ;
                    63 ;       Reg #5  (Read)
0005                64 CPTRG   EQU     PRT91+5
                    65 ;
                    66 ;       Reg #6  Address 0/1 reg. constants
0006                67 ADR01   EQU     PRT91+6
0060                68 DTDL1   EQU     60H     ;Disable major talker & listener
00E0                69 DTDL2   EQU     0E0H    ;Disable minor talker & listener
                    70 ;
                    71 ;       Reg #7  EOS     Character Register
0007                72 EOSR    EQU     PRT91+7
                    73 ;
                    74 ;
                    75 ;       8292    CONTROL VALUES
                    76 ;
                    77 ;
                    78 ;
0068                79 PRT92   EQU     PRT91+8 ;8292 Base Port # (CS7)
                    80 ;
0068                81 INTMR   EQU     PRT92+0 ;92 INTRP Mask Req
00A0                82 INTM    EQU     0A0H    ;TCI
                    83 ;
0068                84 ERRM    EQU     PRT92+0 ;92 Error Mask Req
0001                85 TOUT1   EQU     01      ;92 Time Out for Pass Control
0002                86 TOUT2   EQU     02      ;92 Time Out for Standby
0004                87 TOUT3   EQU     04      ;92 Time Out for Take Control Sync
0068                88 EVREG   EQU     PRT92+0 ;92 Event Counter Pseudo Reg
0068                89 TOREG   EQU     PRT92+0 ;92 Time Out Pseudo Reg
                    90 ;
0069                91 CMD92   EQU     PRT92+1 ;92 Command Register
                    92 ;
0069                93 INTST   EQU     PRT92+1 ;92 Interrupt Status Reg
0010                94 EVBIT   EQU     10H     ;Event Counter Bit
0002                95 IBFBT   EQU     02      ;Input Buffer Full Bit
0020                96 SRQBT   EQU     20H     ;Seq bit
                    97 ;
0068                98 ERFLG   EQU     PRT92+0 ;92 Error Flag Pseudo Reg
0068                99 CLRST   EQU     PRT92+0 ;92 Controller Status Pseudo Reg
0068               100 BUSST   EQU     PRT92+0 ;92 GPIB (Bus) Status Pseudo Reg
0068               101 EVCST   EQU     PRT92+0 ;92 Event Counter Status Pseudo Reg
0068               102 TOST    EQU     PRT92+0 ;92 Time Out Status Pseudo Reg
                   103 ;
                   104 ;       8292    OPERATION COMMANDS
                   105 ;
                   106 ;
00F0               107 SPCNI   EQU     0F0H    ;Stop Counter Interrupts
00F1               108 GIDL    EQU     0F1H    ;Go to idle
00F2               109 RSET    EQU     0F2H    ;Reset
00F3               110 RSTI    EQU     0F3H    ;Reset Interrupts
00F4               111 GSEC    EQU     0F4H    ;Goto standby, enable counting
00F5               112 EXPP    EQU     0F5H    ;Execute parallel poll
00F6               113 GTSB    EQU     0F6H    ;Goto standby
00F7               114 SLOC    EQU     0F7H    ;Set local mode
00F8               115 SREM    EQU     0F8H    ;Set interface to remote
00F9               116 ABORT   EQU     0F9H    ;Abort all operation, clear interface
00FA               117 TCNTR   EQU     0FAH    ;Take control (Receive control)
00FC               118 TCASY   EQU     0FCH    ;Take control asyncronously
00FD               119 TCSY    EQU     0FDH    ;Take control syncronously
00FE               120 STCNI   EQU     0FEH    ;Start counter interrupts
                   121 ;
                   122 ;
```

231324-31

3

```
                    123 ;       8292    UTILITY COMMANDS
                    124 ;
                    125 ;
00E1                126 WOUT    EQU     0E1H    ;Write to timeout reg
00E2                127 WEVC    EQU     0E2H    ;Write to event counter
00E3                128 REVC    EQU     0E3H    ;Read event counter status
00E4                129 RERF    EQU     0E4H    ;Read error flag reg
00E5                130 RINM    EQU     0E5H    ;Read interrupt mask reg
00E6                131 RCST    EQU     0E6H    ;Read controller status reg
00E7                132 RBST    EQU     0E7H    ;Read GPIB Bus status reg
00E9                133 RTOUT   EQU     0E9H    ;Read timeout status reg
00EA                134 RERM    EQU     0EAH    ;Read error mask reg
000B                135 IACK    EQU     0BH     ;Interrupt Acknowledge
                    136 ;
                    137 ;
                    138 ;       PORT F BIT ASSIGNMENTS
                    139 ;
                    140 ;
                    141 ;
006F                142 PRTF    EQU     PRT91+0FH       ;ZT7488 port 6F for interrupts
0002                143 TCIF    EQU     02H     ;Task complete interrupt
0004                144 SPIF    EQU     04H     ;Special interrupt
0008                145 OBFF    EQU     08H     ;92 Output (to CPU) Buffer full
0010                146 IBFF    EQU     10H     ;92 Input (from CPU) Buffer empty
0001                147 BOF     EQU     01H     ;91 Int line (BO in this case)
                    148 ;
                    149 ;       GPIB MESSAGES (COMMANDS)
                    150 ;
0001                151 MDA     EQU     1       ;My device address is 1
0041                152 MTA     EQU     MDA+40H ;My talk address is 1 ("A")
0021                153 MLA     EQU     MDA+20H ;My listen address is 1 ("!")
003F                154 UNL     EQU     3FH     ;Universal unlisten
0008                155 GET     EQU     08      ;Group Execute Trigger
0004                156 SDC     EQU     04H     ;Device Clear
0018                157 SPE     EQU     18H     ;Serial poll enable
0019                158 SPD     EQU     19H     ;Serial poll disable
0005                159 PPC     EQU     05      ;Parallel poll configure
0070                160 PPD     EQU     70H     ;Parallel poll disable
0060                161 PPE     EQU     60H     ;Parallel poll disable
0015                162 PPU     EQU     15H     ;Parallel poll unconfigured
0009                163 TCT     EQU     09      ;Take control (pass control)
                    164
                    165 ;       MACRO DEFINITIONS
                    166 ;
                    167 ;
                    168 ;
                    169 SETF    MACRO           ;Sets flags on A register
                    170         ORA     A
                    171         ENDM
                    172 ;
                    173 WAITO   MACRO           ;Wait for last 91 byte to be done
                    174         LOCAL   WAITL
                    175 WAITL:  IN      INT1    ;Get Int1 status
                    176         ANI     BOM     ;Check for byte out
                    177         JZ      WAITL   ;If not, try again
                    178         ENDM            ;until it is
                    179 ;
                    180 ;
                    181 WAITI   MACRO           ;Wait for 91 byte to be input
                    182         LOCAL   WAITL
                    183 WAITL:  IN      INT1    ;Get INT1 status
                    184         MOV     B,A     ;Save status in B
                    185         ANI     BIM     ;Check for byte in
                    186         JZ      WAITL   ;If not, just try again
                    187         ENDM            ;until it is
                    188 ;
                    189 WAITX   MACRO           ;Wait for 92's TCI to go false
                    190         LOCAL   WAITL
                    191 WAITL:  IN      PRTF
                    192         ANI     TCIF
                    193         JNZ     WAITL
                    194         ENDM
                    195 ;
```

231324-32

```
                        196 WAITT   MACRO
                        197         LOCAL   WAITL
                        198 WAITL:  IN      PRTF    ;Get task complete int,etc.
                        199         ANI     TCIF    ;Mask it
                        200         JZ      WAITL   ;Wait for task to be complete
                        201         ENDM
                        202
                        203 RANGE   MACRO   LOWER,UPPER,LABEL
                        204                         ;Checks for value in range
                        205                         ;branches to label if not
                        206                         ;in range. Falls through if
                        207                         ;lower <= ( (H)(L) ) <= upper.
                        208                         ;Get next byte.
                        209         MOV     A,M
                        210         CPI     LOWER
                        211         JM      LABEL
                        212         CPI     UPPER+1
                        213         JP      LABEL
                        214         ENDM
                        215 ;
                        216 CLRA    MACRO
                        217         XRA     A       ;A XOR A =0
                        218         ENDM
                        219 ;
                        220 ;       All of the following routines have these common
                        221 ;       assumptions about the state of the 8291 & 9292 upon entry
                        222 ;       to the routine and will exit the routine in an identical state.
                        223 ;
                        224 ;
                        225 ;               8291:   BO is or has been set,
                        226 ;                       All interrupts are masked off
                        227 ;                       TON mode, not LA
                        228 ;                       No holdoffs in effect or enabled
                        229 ;                       No holdoffs waiting for finish command
                        230 ;
                        231 ;               8292:   ATN asserted (active controller)
                        232 ;                       note: RCTL is an exception--- it expects
                        233 ;                       to not be active controller
                        234 ;                       Any previous task is complete & 92 is
                        235 ;                       ready to receive next command.
                        236 ;               8085:   Pointer registers (DE,HL) end one
                        237 ;                       beyond last legal entry
                        238 ;*****************************************************
                        239 ;
                        240 ;
                        241 ;       INITIALIZATION ROUTINE
                        242 ;
                        243 ;INPUTS:          None
                        244 ;OUTPUTS:         None
                        245 ;CALLS:           None
                        246 ;DESTROYS:        A,F
                        247 ;
1000 3EA0               248 INIT:   MVI     A,INTM  ;Enable TCI
1002 D368               249         OUT     INTMR   ;Output to 92's intr. mask reg
1004 3E60               250         MVI     A,DTDL1 ;Disable major talker/listener
1006 D366               251         OUT     ADR01
1008 3EE0               252         MVI     A,DTDL2 ;Disable minor talker/listener
100A D366               253         OUT     ADR01
100C 3E80               254         MVI     A,TON   ;Talk only mode
100E D364               255         OUT     ADRMD
1010 3E23               256         MVI     A,CLKRT ;3 MHZ for delay timer
1012 D365               257         OUT     AUXMD
                        258         CLRA
1014 AF                 259+        XRA     A       ;A XOR A =0
1015 D361               260         OUT     INT1
1017 D362               261         OUT     INT2    ;Disable all 91 mask bits
1019 D365               262         OUT     AUXMD   ;Immediate execute PON
101B C9                 263         RET
                        264 ;
                        265 ;*****************************************************
                        266 ;
                        267 ;
                        268 ;       SEND ROUTINE
                        269 ;
```

231324-33

```
                        270 ;
                        271 ;
                        272 ;       INPUTS:         HL listener list pointer
                        273 ;                       DE data buffer pointer
                        274 ;                       C  count-- 0 will cause no data to be sent
                        275 ;                       b  EOS character-- software detected
                        276 ;       OUTPUTS:        none
                        277 ;       CALLS:          none
                        278 ;       DESTROYS:       A, C, DE, HL, F
                        279 ;
                        280 ;
                        281 ;
101C 3E41               282 SEND:   MVI     A,MTA   ;Send MTA to turn off any
101E D360               283         OUT     DOUT    ;previous talker
                        284         WAITO
1020 DB61               285+??0001: IN      INT1    ;Get Intl status
1022 E602               286+        ANI     BOM     ;Check for byte out
1024 CA2010             287+        JZ      ??0001  ;If not, try again
1027 3E3F               288         MVI     A,UNL   ;Send universal unlisten
1029 D360               289         OUT     DOUT    ;to stop previous listeners
102B 78                 290         MOV     A,B     ;Get EOS character
102C D367               291         OUT     EOSR    ;Output it to 8291
                        292                         ;while listener.....
                        293 SEND1:  RANGE   20H,3EH,SEND2   ;Check next listen address
                        294+                        ;Checks for value in range
                        295+                        ;branches to label if not
                        296+                        ;in range. Falls through if
                        297+                        ;lower <= ( (H)(L) ) <= upper.
                        298+                        ;Get next byte.
102E 7E                 299+        MOV     A,M
102F FE20               300+        CPI     20H
1031 FA4710             301+        JM      SEND2
1034 FE3F               302+        CPI     3EH+1
1036 F24710             303+        JP      SEND2
                        304         WAITO                   ;Wait for previous listener sent
1039 DB61               305+??0002: IN      INT1    ;Get Intl status
103B E602               306+        ANI     BOM     ;Check for byte out
103D CA3910             307+        JZ      ??0002  ;If not, try again
1040 7E                 308         MOV     A,M             ;Get this listener
1041 D360               309         OUT     DOUT            ;Output to GPIB
1043 23                 310         INX     H               ;Increment listener list pointer
1044 C32E10             311         JMP     SEND1           ;Loop till non-valid listener
                        312                                 ;Enable 91 ending conditions
                        313 SEND2:  WAITO                   ;Wait for lstn addr accepted
1047 DB61               314+??0003: IN      INT1    ;Get Intl status
1049 E602               315+        ANI     BOM     ;Check for byte out
104B CA4710             316+        JZ      ??0003  ;If not, try again
                        317                         ;WAITO required for early versions
                        318                         ;of 8292 to avoid GTSB before DAC
104E 3EF6               319         MVI     A,GTSB  ;Goto standby
1050 D369               320         OUT     CMD92   ;
1052 3E88               321         MVI     A,AXRA+EOIS     ;Send EOI with EOS character
1054 D365               322         OUT     AUXMD
                        323         WAITX               ;Wait for TCI to go false
1056 DB6F               324+??0004: IN      PRTF
1058 E602               325+        ANI     TCIF
105A C25610             326+        JNZ     ??0004
                        327         WAITT               ;Wait for TCI on GTSB
105D DB6F               328+??0005: IN      PRTF    ;Get task complete int,etc.
105F E602               329+        ANI     TCIF    ;Mask it
1061 CA5D10             330+        JZ      ??0005  ;Wait for task to be complete
                        331
                        332 ;       delete next 3 instructions to make count of 0=256
                        333 ;
1064 79                 334         MOV     A,C     ;Get count
                        335         SETF            ;Set flags
1065 B7                 336+        ORA     A
1066 CA8810             337         JZ      SEND6   ;If count=0, send no data
1069 1A                 338 SEND3:  LDAX    D       ;Get data byte
106A D360               339         OUT     DOUT    ;Output to GPIB
106C B8                 340         CMP     B       ;Test EOS ...this is faster
                        341                         ;and uses less code than using
                        342                         ;91's END or EOI bits
```

231324-34

```
106D CA7F10          343          JZ      SEND5   ;If char = EOS , go finish
                     344 SEND4:   WAITO
1070 DB61            345+??0006:  IN      INT1    ;Get Intl status
1072 E602            346+         ANI     BOM     ;Check for byte out
1074 CA7010          347+         JZ      ??0006  ;If not, try again
1077 13              348          INX     D       ;Increment buffer pointer
1078 0D              349          DCR     C       ;Decrement count
1079 C26910          350          JNZ     SEND3   ;If count < > 0, go send
107C C38810          351          JMP     SEND6   ;Else go finish
107F 13              352 SEND5:   INX     D       ;for consistency
1080 0D              353          DCR     C       ; "     "
                     354          WAITO
                                                  ;This ensures that the standard entry
1081 DB61            355+??0007:  IN      INT1    ;Get Intl status
1083 E602            356+         ANI     BOM     ;Check for byte out
1085 CA8110          357+         JZ      ??0007  ;If not, try again
                     358                          ;assumptions for the next subroutine are met
1088 3EFD            359 SEND6:   MVI     A,TCSY  ;Take control syncronously
108A D369            360          OUT     CMD92
108C 3E80            361          MVI     A,AXRA  ;Reset send EOI on EOS
108E D365            362          OUT     AUXMD
                     363          WAITX           ;Wait for TCI false
1090 DB6F            364+??0008:  IN      PRTF
1092 E602            365+         ANI     TCIF
1094 C29010          366+         JNZ     ??0008
                     367          WAITT           ;Wait for TCI
1097 DB6F            368+??0009:  IN      PRTF    ;Get task complete int,etc.
1099 E602            369+         ANI     TCIF    ;Mask it
109B CA9710          370+         JZ      ??0009  ;Wait for task to be complete
109E C9              371          RET
                     372 ;*****************************************************************
                     373 ;
                     374 ;        RECEIVE ROUTINE
                     375 ;
                     376 ;
                     377 ;INPUT:          HL talker pointer
                     378 ;               DE data buffer pointer
                     379 ;               C  count (max buffer size) 0 implies 256
                     380 ;               B  EOS character
                     381 ;OUTPUT:         Fills buffer pointed at by DE
                     382 ;CALLS:          None
                     383 ;DESTROYS:       A, BC, DE, HL, F
                     384 ;
                     385 ;RETURNS:        A=0 normal termination--EOS detected
                     386 ;               A=40 Error--- count overrun
                     387 ;               A<40 or A>5EH Error--- bad talk address
                     388 ;
                     389 ;
109F 78              390 RECV:    MOV     A,B     ;Get EOS character
10A0 D367            391          OUT     EOSR    ;Output it to 91
                     392          RANGE   40H,5EH,RECV6
                     393+                         ;Checks for value in range
                     394+                         ;branches to label if not
                     395+                         ;in range. Falls through if
                     396+                         ;lower <= ( (H)(L) ) <= upper.
                     397+                         ;Get next byte.
10A2 7E              398+         MOV     A,M
10A3 FE40            399+         CPI     40H
10A5 FA3911          400+         JM      RECV6
10A8 FE5F            401+         CPI     5EH+1
10AA F23911          402+         JP      RECV6
                     403                          ;valid if 40H<= talk <=5EH
10AD D360            404          OUT     DOUT    ;Output talker to GPIB
10AF 23              405          INX     H       ;Incr pointer for consistency
                     406          WAITO
10B0 DB61            407+??0010:  IN      INT1    ;Get Intl status
10B2 E602            408+         ANI     BOM     ;Check for byte out
10B4 CAB010          409+         JZ      ??0010  ;If not, try again
10B7 3E3F            410          MVI     A,UNL   ;Stop other listeners
10B9 D360            411          OUT     DOUT
                     412          WAITO
10BB DB61            413+??0011:  IN      INT1    ;Get Intl status
10BD E602            414+         ANI     BOM     ;Check for byte out
10BF CABB10          415+         JZ      ??0011  ;If not, try again
```

231324-35

3

```
10C2 3E21          416          MVI     A,MLA     ;For completeness
10C4 D350          417          OUT     DOUT
10C6 3E86          418          MVI     A,AXRA+HOEND+EDEOS      ;End when
10C8 D365          419          OUT     AUXMD     ;EOS or EOI & Holdoff
                   420          WAITO
10CA DB61          421+??0012:   IN      INT1      ;Get Intl status
10CC E602          422+         ANI     BOM       ;Check for byte out
10CE CACA10        423+         JZ      ??0012    ;If not, try again
10D1 3E40          424          MVI     A,LON     ;Listen only
10D3 D364          425          OUT     ADRMD
                   426          CLRA               ;Immediate XEQ PON
10D5 AF            427+         XRA     A         ;A XOR A =0
10D6 D365          428          OUT     AUXMD
10D8 3EF6          429          MVI     A,GTSB    ;Goto standby
10DA D359          430          OUT     CMD92
                   431          WAITX              ;Wait for TCI=0
10DC DB6F          432+??0013:  IN      PRTF
10DE E602          433+         ANI     TCIF
10E0 C2DC10        434+         JNZ     ??0013
                   435          WAITT              ;Wait for TCI=1
10E3 DB6F          436+??0014:  IN      PRTF      ;Get task complete int,etc.
10E5 E602          437+         ANI     TCIF      ;Mask it
10EA DB61          439 RECV1:   IN      INT1      ;Get 91 Int status (END &/or BI)
10EC 47            440          MOV     B,A       ;Save it in B for BI check later
10ED E610          441          ANI     ENDMK     ;Check for EOS or EOI
10EF C20511        442          JNZ     RECV2     ;Yes end--- go wait for BI
10F2 78            443          MOV     A,B       ;NO, retrieve status &
10F3 E601          444          ANI     BIM       ;check for BI
10F5 CAEA10        445          JZ      RECV1     ;NO, go wait for either END or BI
10F8 DB60          446          IN      DIN       ;YES, BI--- get data
10FA 12            447          STAX    D         ;Store it in buffer
10FB 13            448          INX     D         ;Increment buffer pointer
10FC 0D            449          DCR     C         ;Decrement counter
10FD C2EA10        450          JNZ     RECV1     ;If count < > 0 go back & wait
1100 0640          451          MVI     B,40H     ;Else set error indicator
1102 C31711        452          JMP     RECV5     ;And go take control
                   453 ;
1105 78            454 RECV2:   MOV     A,B       ;Retreive status
1106 E601          455 RECV3:   ANI     BIM       ;Check for BI
1108 C21011        456          JNZ     RECV4     ;If BI then go input data
110B DB61          457          IN      INT1      ;Else wait for last BI
110D C30611        458          JMP     RECV3     ;In loop
1110 DB60          459 RECV4:   IN      DIN       ;Get data byte
1112 12            460          STAX    D         ;Store it in buffer
1113 13            461          INX     D         ;Incr data pointer
1114 0D            462          DCR     C         ;Decrement count, but ignore it
1115 0600          463          MVI     B,0       ;Set normal completion indicators
                   464 ;
1117 3EFD          465 RECV5:   MVI     A,TCSY    ;Take control synchronously
1119 D369          466          OUT     CMD92
                   467          WAITX              ;Wait for TCI=0 (7 tcy)
111B DB6F          468+??0015:  IN      PRTF
111D E602          469+         ANI     TCIF
111F C21B11        470+         JNZ     ??0015
                   471          WAITT              ;Wait for TCI=1
1122 DB6F          472+??0016:  IN      PRTF      ;Get task complete int,etc.
1124 E602          473+         ANI     TCIF      ;Mask it
1126 CA2211        474+         JZ      ??0016    ;Wait for task to be complete
                   475 ;
                   476 ;if timeout 3 is to be checked, the above WAITT should
                   477 ;be omitted & the appropriate code to look for TCI or
                   478 ;TOUT3 inserted here.
                   479 ;
1129 3E80          480          MVI     A,AXRA    ;Pattern to clear 91 END conditions
112B D365          481          OUT     AUXMD     ;
112D 3E80          482          MVI     A,TON     ;This bit pattern already in "A"
112F D364          483          OUT     ADRMD     ;Output TON
1131 3E03          484          MVI     A,FNHSK   ;Finish handshake
1133 D365          485          OUT     AUXMD
                   486          CLRA
1135 AF            487+         XRA     A         ;A XOR A =0
1136 D365          488          OUT     AUXMD     ;Immediate execute PON-Reset LON
1138 78            489          MOV     A,B       ;Get completion character
1139 C9            490 RECV6:   RET
```

231324-36

```
                    491 ;
                    492 ;*************************************************************
                    493 ;       XFER ROUTINE
                    494 ;
                    495 ;
                    496 ;INPUTS:          HL device list pointer
                    497 ;                 B  EOS character
                    498 ;OUTPUTS:         None
                    499 ;CALLS:           None
                    500 ;DESTROYS:        A, HL, F
                    501 ;RETURNS:         A=0 normal, A < > 0 bad talker
                    502 ;
                    503 ;
                    504 ;NOTE:            XFER will not work if the talker
                    505 ;                 uses EOI to terminate the transfer.
                    506 ;                 Intel will be making hardware
                    507 ;                 modifications to the 8291 that will
                    508 ;                 correct this problem. Until that time,
                    509 ;                 only EOS may be used without possible
                    510 ;                 loss of the last data byte transfered.
                    511 XFER:    RANGE    40H,5EH,XFER4   ;Check for valid talker
                    512+                                 ;Checks for value in range
                    513+                                 ;branches to label if not
                    514+                                 ;in range. Falls through if
                    515+                                 ;lower <= ( (H)(L) ) <= upper.
                    516+                                 ;Get next byte.
113A 7E             517+            MOV      A,M
113B FE40           518+            CPI      40H
113D FABB11         519+            JM       XFER4
1140 FE5F           520+            CPI      5EH+1
1142 F2BB11         521+            JP       XFER4
1145 D350           522             OUT      DOUT       ;Send it to GPIB
1147 23             523             INX      H          ;Incr pointer
                    524             WAITO
1148 DB61           525+??0017:      IN      INT1       ;Get Intl status
114A E602           526+            ANI      BOM        ;Check for byte out
114C CA4811         527+            JZ       ??0017     ;If not, try again
114F 3E3F           528             MVI      A,UNL      ;Universal unlisten
1151 D360           529             OUT      DOUT
                    530 XFER1:  RANGE    20H,3EH,XFER2   ;Check for valid listener
                    531+                                 ;Checks for value in range
                    532+                                 ;branches to label if not
                    533+                                 ;in range. Falls through if
                    534+                                 ;lower <= ( (H)(L) ) <= upper.
                    535+                                 ;Get next byte.
1153 7E             536+            MOV      A,M
1154 FE20           537+            CPI      20H
1156 FA6C11         538+            JM       XFER2
1159 FE3F           539+            CPI      3EH+1
115B F26C11         540+            JP       XFER2
                    541             WAITO
115E DB61           542+??0018:      IN      INT1       ;Get Intl status
1160 E602           543+            ANI      BOM        ;Check for byte out
1162 CA5E11         544+            JZ       ??0018     ;If not, try again
1165 7E             545             MOV      A,M        ;Get listener
1166 D360           546             OUT      DOUT
1168 23             547             INX      H          ;Incr pointer
1169 C35311         548             JMP      XFER1      ;Loop until non-valid listener
                    549 XFER2:  WAITO
116C DB61           550+??0019:      IN      INT1       ;Get Intl status
116E E602           551+            ANI      BOM        ;Check for byte out
1170 CA6C11         552+            JZ       ??0019     ;If not, try again
1173 3E87           553             MVI      A,AXRA+CAHCY+EDEOS      ;Invisible handshake
1175 D365           554             OUT      AUXMD      ;Continuous AH mode
1177 3E40           555             MVI      A,LON      ;Listen only
1179 D364           556             OUT      ADRMD
                    557             CLRA
117B AF             558+            XRA      A          ;A XOR A =0
117C D365           559             OUT      AUXMD      ;Immed. XEQ PON
117E 78             560             MOV      A,B        ;Get EOS
117F D367           561             OUT      EOSR       ;Output it to 91
1181 3EF6           562             MVI      A,GTSB     ;Go to standby
1183 D369           563             OUT      CMD92
```

231324-37

```
                        564             WAITX
1185 DB6F               565+??0020:     IN      PRTF
1187 E602               566+            ANI     TCIF
1189 C28511             567+            JNZ     ??0020
                        568             WAITT           ;Wait for TCS
118C DB6F               569+??0021:     IN      PRTF    ;Get task complete int,etc.
118E E602               570+            ANI     TCIF    ;Mask it
1190 CA8C11             571+            JZ      ??0021  ;Wait for task to be complete
1193 DB61               572 XFER3:      IN      INT1    ;Get END status bit
1195 E610               573             ANI     ENDMK   ;Mask it
1197 CA9311             574             JZ      XFER3
119A 3EFD               575             MVI     A,TCSY  ;Take control syncronously
119C D369               576             OUT     CMD92
                        577             WAITX
119E DB6F               578+??0022:     IN      PRTF
11A0 E602               579+            ANI     TCIF
11A2 C29E11             580+            JNZ     ??0022
                        581             WAITT           ;Wait for TCI
11A5 DB6F               582+??0023:     IN      PRTF    ;Get task complete int,etc.
11A7 E602               583+            ANI     TCIF    ;Mask it
11A9 CAA511             584+            JZ      ??0023  ;Wait for task to be complete
11AC 3E80               585             MVI     A,AXRA  ;Not cont AH or END on EOS
11AE D365               586             OUT     AUXMD
11B0 3E03               587             MVI     A,FNHSK ;Finish handshake
11B2 D365               588             OUT     AUXMD
11B4 3E80               589             MVI     A,TON   ;Talk only
11B6 D364               590             OUT     ADRMD
                        591             CLRA            ;Normal return A=0
11B8 AF                 592+            XRA     A       ;A XOR A =0
11B9 D365               593             OUT     AUXMD   ;Immediate XEQ PON
11BB C9                 594 XFER4:      RET
                        595 ;
                        596 ;*************************************************
                        597 ;
                        598 ;
                        599 ;       TRIGGER ROUTINE
                        600 ;
                        601 ;
                        602 ;INPUTS:         HL listener list pointer
                        603 ;OUTPUTS:        None
                        604 ;CALLS:          None
                        605 ;DESTROYS:       A, HL, F
                        606 ;
                        607 ;
11BC 3E3F               608 TRIG:       MVI     A,UNL   ;
11BE D360               609             OUT     DOUT    ;Send universal unlisten
                        610 TRIG1:      RANGE   20H,3EH,TRIG2   ;Check for valid listen
                        611+                            ;Checks for value in range
                        612+                            ;branches to label if not
                        613+                            ;in range. Falls through if
                        614+                            ;lower <= ( (H)(L) ) <= upper.
                        615+                            ;Get next byte.
11C0 7E                 616+            MOV     A,M
11C1 FE20               617+            CPI     20H
11C3 FAD911             618+            JM      TRIG2
11C6 FE3F               619+            CPI     3EH+1
11C8 F2D911             620+            JP      TRIG2
                        621             WAITO           ;Wait for UNL to finish
11CB DB61               622+??0024:     IN      INT1    ;Get Intl status
11CD E602               623+            ANI     BOM     ;Check for byte out
11CF CACB11             624+            JZ      ??0024  ;If not, try again
11D2 7E                 625             MOV     A,M     ;Get listener
11D3 D360               626             OUT     DOUT    ;Send Listener to GPIB
11D5 23                 627             INX     H       ;Incr. pointer
11D6 C3C011             628             JMP     TRIG1   ;Loop until non-valid char
                        629 TRIG2:      WAITO           ;Wait for last listen to finish
11D9 DB61               630+??0025:     IN      INT1    ;Get Intl status
11DB E602               631+            ANI     BOM     ;Check for byte out
11DD CAD911             632+            JZ      ??0025  ;If not, try again
11E0 3E08               633             MVI     A,GET   ;Send group execute trigger
11E2 D360               634             OUT     DOUT    ;to all addressed listeners
                        635             WAITO
11E4 DB61               636+??0026:     IN      INT1    ;Get Intl status
11E6 E602               637+            ANI     BOM     ;Check for byte out
```

231324-38

```
11E8 CAE411          638+        JZ      ??0026  ;If not, try again
11EB C9              639         RET
                     640 ;
                     641 ;*****************************************
                     642 ;
                     643 ;DEVICE CLEAR ROUTINE
                     644 ;
                     645 ;
                     646 ;
                     647 ;INPUTS:         HL listener pointer
                     648 ;OUTPUT:         None
                     649 ;CALLS:          None
                     650 ;DESTROYS:       A, HL, F
                     651 ;
11EC 3E3F            652 DCLR:   MVI      A,UNL
11EE D360            653         OUT      DOUT
                     654 DCLR1:  RANGE    20H,3EH,DCLR2
                     655+                         ;Checks for value in range
                     656+                         ;branches to label if not
                     657+                         ;in range. Falls through if
                     658+                         ;lower <= ( (H)(L) ) <= upper.
                     659+                         ;Get next byte.
11F0 7E              660+        MOV      A,M
11F1 FE20            661+        CPI      20H
11F3 FA0912          662+        JM       DCLR2
11F6 FE3F            663+        CPI      3EH+1
11F8 F20912          664+        JP       DCLR2
                     665         WAITO
11FB DB61            666+??0027:  IN      INT1    ;Get Intl status
11FD E602            667+        ANI      BOM     ;Check for byte out
11FF CAFB11          668+        JZ       ??0027  ;If not, try again
1202 7E              669         MOV      A,M
1203 D360            670         OUT      DOUT    ;Send listener to GPIB
1205 23              671         INX      H
1206 C3F011          672         JMP      DCLR1
                     673 DCLR2:  WAITO
1209 DB61            674+??0028:  IN      INT1    ;Get Intl status
120B E602            675+        ANI      BOM     ;Check for byte out
120D CA0912          676+        JZ       ??0028  ;If not, try again
1210 3E04            677         MVI      A,SDC   ;Send device clear
1212 D360            678         OUT      DOUT    ;To all addressed listeners
                     679         WAITO
1214 DB61            680+??0029:  IN      INT1    ;Get Intl status
1216 E602            681+        ANI      BOM     ;Check for byte out
1218 CA1412          682+        JZ       ??0029  ;If not, try again
121B C9              683         RET
                     684 ;
                     685 ;*************************************************
                     686 ;
                     687 ;       SERIAL POLL ROUTINE
                     688 ;
                     689 ;INPUTS:         HL talker list pointer
                     690 ;               DE status buffer pointer
                     691 ;OUTPUTS:        Fills buffer pointed to by DE
                     692 ;CALLS:          None
                     693 ;DESTROYS:       A, BC, DE, HL, F
                     694 ;
121C 3E3F            695 SPOL:   MVI      A,UNL   ;Universal unlisten
121E D360            696         OUT      DOUT
                     697         WAITO
1220 DB61            698+??0030:  IN      INT1    ;Get Intl status
1222 E602            699+        ANI      BOM     ;Check for byte out
1224 CA2012          700+        JZ       ??0030  ;If not, try again
1227 3E21            701         MVI      A,MLA   ;My listen address
1229 D360            702         OUT      DOUT
                     703         WAITO
122B DB61            704+??0031:  IN      INT1    ;Get Intl status
122D E602            705+        ANI      BOM     ;Check for byte out
122F CA2B12          706+        JZ       ??0031  ;If not, try again
1232 3E18            707         MVI      A,SPE   ;Serial poll enable
1234 D360            708         OUT      DOUT    ;To be formal about it
                     709         WAITO
1236 DB61            710+??0032:  IN      INT1    ;Get Intl status
```

231324-39

3

```
1238 E602          711+          ANI     BOM       ;Check for byte out
123A CA3612        712+          JZ      ??0032    ;If not, try again
                   713 SPOL1:    RANGE   40H,5EH,SPOL2   ;Check for valid talker
                   714+                                  ;Checks for value in range
                   715+                                  ;branches to label if not
                   716+                                  ;in range. Falls through if
                   717+                                  ;lower <= ( (H)(L) ) <= upper.
                   718+                                  ;Get next byte.
123D 7E            719+          MOV     A,M
123E FE40          720+          CPI     40H
1240 FA9412        721+          JM      SPOL2
1243 FE5F          722+          CPI     5EH+1
1245 F29412        723+          JP      SPOL2
1248 7E            724           MOV     A,M       ;Get talker
1249 D360          725           OUT     DOUT      ;Send to GPIB
124B 23            726           INX     H         ;Incr talker list pointer
124C 3E40          727           MVI     A,LON     ;Listen only
124E D364          728           OUT     ADRMD
                   729           WAITO             ;Wait for talk address to complete
1250 DB61          730+??0033:    IN     INT1      ;Get Intl status
1252 E602          731+          ANI     BOM       ;Check for byte out
1254 CA5012        732+          JZ      ??0033    ;If not, try again
                   733           CLRA              ;Pattern for immediate XEQ PON
1257 AF            734+          XRA     A         ;A XOR A =0
1258 D365          735           OUT     AUXMD
125A 3EF6          736           MVI     A,GTSB    ;Goto standby
125C D369          737           OUT     CMD92
                   738           WAITX             ;Wait for TCI false
125E DB6F          739+??0034:   IN      PRTF
1260 E602          740+          ANI     TCIF
1262 C25E12        741+          JNZ     ??0034
                   742           WAITT             ;Wait for TCI
1265 DB6F          743+??0035:   IN      PRTF      ;Get task complete int,etc.
1267 E602          744+          ANI     TCIF      ;Mask it
1269 CA6512        745+          JZ      ??0035    ;Wait for task to be complete
                   746           WAITI             ;Wait for status byte input
126C DB61          747+??0036:   IN      INT1      ;Get INTl status
126E 47            748+          MOV     B,A       ;Save status in B
126F E601          749+          ANI     BIM       ;Check for byte in
1271 CA6C12        750+          JZ      ??0036    ;If not, just try again
1274 3EFD          751           MVI     A,TCSY    ;Take control sync
1276 D359          752           OUT     CMD92
                   753           WAITX             ;Wait for TCI false
1278 DB6F          754+??0037:   IN      PRTF
127A E602          755+          ANI     TCIF
127C C27812        756+          JNZ     ??0037
                   757           WAITT             ;Wait for TCI
127F DB6F          758+??0038:   IN      PRTF      ;Get task complete int,etc.
1281 E602          759+          ANI     TCIF      ;Mask it
1283 CA7F12        760+          JZ      ??0038    ;Wait for task to be complete
1286 DB60          761           IN      DIN       ;Get serial poll status byte
1288 12            762           STAX    D         ;Store it in buffer
1289 13            763           INX     D         ;Incr pointer
128A 3E80          764           MVI     A,TON     ;Talk only for controller
128C D364          765           OUT     ADRMD     ;
                   766           CLRA
128E AF            767+          XRA     A         ;A XOR A =0
128F D365          768           OUT     AUXMD     ;Immeditate XEQ PON
                   769                             ;CLR LA
1291 C33D12        770           JMP     SPOL1     ;Go on to next device on list
                   771 ;
1294 3E19          772 SPOL2:    MVI     A,SPD     ;Serial poll disable
1296 D360          773           OUT     DOUT      ;We know BO was set (WAITO above)
                   774           WAITO
1298 DB61          775+??0039:    IN     INT1      ;Get Intl status
129A E602          776+          ANI     BOM       ;Check for byte out
129C CA9812        777+          JZ      ??0039    ;If not, try again
                   778           CLRA
129F AF            779+          XRA     A         ;A XOR A =0
12A0 D365          780           OUT     AUXMD     ;Immediate XEQ PON to clear LA
12A2 C9            781           RET
                   782 ;
                   783 ;*****************************************************
                   784 ;
```

231324-40

```
                    785 ;       PARALLEL POLL ENABLE ROUTINE
                    786 ;
                    787 ;INPUTS:         HL listener list pointer
                    788 ;                DE configuration byte pointer
                    789 ;OUTPUTS:        None
                    790 ;CALLS:          None
                    791 ;DESTROYS:       A, DE, HL, F
                    792 ;
                    793 ;
12A3 3E3F           794 PPEN:   MVI     A,UNL   ;Universal unlisten
12A5 D360           795         OUT     DOUT
                    796 PPEN1:  RANGE   20H,3EH,PPEN2   ;Check for valid listener
                    797+                        ;Checks for value in range
                    798+                        ;branches to label if not
                    799+                        ;in range. Falls through if
                    800+                        ;lower <= ( (H)(L) ) <= upper.
                    801+                        ;Get next byte.
12A7 7E             802+        MOV     A,M
12A8 FE20           803+        CPI     20H
12AA FAD812         804+        JM      PPEN2
12AD FE3F           805+        CPI     3EH+1
12AF F2D812         806+        JP      PPEN2
                    807         WAITO           ;Valid wait 91 data out req
12B2 DB61           808+??0040:  IN     INT1    ;Get Intl status
12B4 E602           809+        ANI     BOM     ;Check for byte out
12B6 CAB212         810+        JZ      ??0040  ;If not, try again
12B9 7E             811         MOV     A,M     ;Get listener
12BA D360           812         OUT     DOUT
                    813         WAITO
12BC DB61           814+??0041:  IN     INT1    ;Get Intl status
12BE E602           815+        ANI     BOM     ;Check for byte out
12C0 CABC12         816+        JZ      ??0041  ;If not, try again
12C3 3E05           817         MVI     A,PPC   ;Parallel poll configure
12C5 D360           818         OUT     DOUT
                    819         WAITO
12C7 DB61           820+??0042:  IN     INT1    ;Get Intl status
12C9 E602           821+        ANI     BOM     ;Check for byte out
12CB CAC712         822+        JZ      ??0042  ;If not, try again
12CE 1A             823         LDAX    D       ;Get matching configuration byte
12CF F660           824         ORI     PPE     ;Merge with parallel poll enable
12D1 D360           825         OUT     DOUT
12D3 23             826         INX     H       ;Incr pointers
12D4 13             827         INX     D
12D5 C3A712         828         JMP     PPEN1   ;Loop until invalid listener char
                    829 PPEN2:  WAITO
12D8 DB61           830+??0043:  IN     INT1    ;Get Intl status
12DA E602           831+        ANI     BOM     ;Check for byte out
12DC CAD812         832+        JZ      ??0043  ;If not, try again
12DF C9             833         RET
                    834 ;
                    835 ;PARALLEL POLL DISABLE ROUTINE
                    836 ;
                    837 ;INPUTS:         HL listener list pointer
                    838 ;OUTPUTS:        None
                    839 ;CALLS:          None
                    840 ;DESTROYS:       A, HL, F
                    841 ;
12E0 3E3F           842 PPDS:   MVI     A,UNL   ;Universal unlisten
12E2 D360           843         OUT     DOUT
                    844 PPDS1:  RANGE   20H,3EH,PPDS2   ;Check for valid listener
                    845+                        ;Checks for value in range
                    846+                        ;branches to label if not
                    847+                        ;in range. Falls through if
                    848+                        ;lower <= ( (H)(L) ) <= upper.
                    849+                        ;Get next byte.
12E4 7E             850+        MOV     A,M
12E5 FE20           851+        CPI     20H
12E7 FAFD12         852+        JM      PPDS2
12EA FE3F           853+        CPI     3EH+1
12EC F2FD12         854+        JP      PPDS2
                    855         WAITO
12EF DB61           856+??0044:  IN     INT1    ;Get Intl status
12F1 E602           857+        ANI     BOM     ;Check for byte out
12F3 CAEF12         858+        JZ      ??0044  ;If not, try again
```

231324-41

3

```
12F6 7E          859          MOV     A,M     ;Get listener
12F7 D360        860          OUT     DOUT
12F9 23          861          INX     H       ;Incr pointer
12FA C3E412      862          JMP     PPDS1   ;Loop until invalid listener
                 863 PPDS2:   WAITO
12FD DB61        864+??0045:   IN      INT1    ;Get Intl status
12FF E602        865+         ANI     BOM     ;Check for byte out
1301 CAFD12      866+         JZ      ??0045  ;If not, try again
1304 3E05        867          MVI     A,PPC   ;Parallel poll configure
1306 D360        868          OUT     DOUT
                 869          WAITO
1308 DB61        870+??0046:   IN      INT1    ;Get Intl status
130A E602        871+         ANI     BOM     ;Check for byte out
130C CA0813      872+         JZ      ??0046  ;If not, try again
130F 3E70        873          MVI     A,PPD   ;Parallel poll disable
1311 D360        874          OUT     DOUT
                 875          WAITO
1313 DB61        876+??0047:   IN      INT1    ;Get Intl status
1315 E602        877+         ANI     BOM     ;Check for byte out
1317 CA1313      878+         JZ      ??0047  ;If not, try again
131A C9          879          RET
                 880 ;
                 881 ;        PARALLEL POLL UNCONFIGURE ALL ROUTINE
                 882 ;
                 883 ;
                 884 ;INPUTS:          None
                 885 ;OUTPUTS:         None
                 886 ;CALLS:           None
                 887 ;DESTROYS:        A, F
                 888 ;
131B 3E15        889 PPUN:    MVI     A,PPU   ;Parallel poll unconfigure
131D D360        890          OUT     DOUT
                 891          WAITO
131F DB61        892+??0048:   IN      INT1    ;Get Intl status
1321 E602        893+         ANI     BOM     ;Check for byte out
1323 CA1F13      894+         JZ      ??0048  ;If not, try again
1326 C9          895          RET
                 896 ;
                 897 ;*********************************************
                 898 ;
                 899 ;CONDUCT A PARALLEL POLL
                 900 ;
                 901 ;
                 902 ;INPUTS:          None
                 903 ;OUTPUTS:         None
                 904 ;CALLS:           None
                 905 ;DESTROYS:        A, B, F
                 906 ;RETURNS:         A= parallel poll status byte
                 907 ;
1327 3E40        908 PPOL:    MVI     A,LON   ;Listen only
1329 D364        909          OUT     ADRMD
                 910          CLRA            ;Immediate XEQ PON
132B AF          911+         XRA     A       ;A XOR A =0
132C D365        912          OUT     AUXMD   ;Reset TON
132E 3EF5        913          MVI     A,EXPP  ;Execute parallel poll
1330 D369        914          OUT     CMD92
                 915          WAITI           ;Wait for completion= BI on 91
1332 DB61        916+??0049:   IN      INT1    ;Get INTl status
1334 47          917+         MOV     B,A     ;Save status in B
1335 E601        918+         ANI     BIM     ;Check for byte in
1337 CA3213      919+         JZ      ??0049  ;If not, just try again
133A 3E80        920          MVI     A,TON   ;Talk only
133C D364        921          OUT     ADRMD
                 922          CLRA            ;Immediate XEQ PON
133E AF          923+         XRA     A       ;A XOR A =0
133F D365        924          OUT     AUXMD   ;Reset LON
1341 DB60        925          IN      DIN     ;Get PP byte
1343 C9          926          RET
                 927 ;
                 928 ;*********************************************
                 929 ;PASS CONTROL ROUTINE
                 930 ;
                 931 ;INPUTS:          HL pointer to talker
                 932 ;OUTPUTS:         None
```

231324-42

```
                        933 ;CALLS:         None
                        934 ;DESTROYS:      A, HL, F
                        935 PCTL:   RANGE   40H,5EH,PCTL1   ;Is it a valid talker ?
                        936+                               ;Checks for value in range
                        937+                               ;branches to label if not
                        938+                               ;in range. Falls through if
                        939+                               ;lower <= ( (H)(L) ) <= upper.
                        940+                               ;Get next byte.
1344 7E                 941+        MOV     A,M
1345 FE40               942+        CPI     40H
1347 FA8A13             943+        JM      PCTL1
134A FE5F               944+        CPI     5EH+1
134C F28A13             945+        JP      PCTL1
134F FE41               946         CPI     MTA         ;Is it my talker address
1351 CA8A13             947         JZ      PCTL1       ;Yes, just return
1354 D360               948         OUT     DOUT        ;Send on GPIB
                        949         WAITO
1356 DB61               950+??0050:  IN      INT1        ;Get Intl status
1358 E602               951+        ANI     BOM         ;Check for byte out
135A CA5613             952+        JZ      ??0050      ;If not, try again
135D 3E09               953         MVI     A,TCT       ;Take control message
135F D360               954         OUT     DOUT
                        955         WAITO
1361 DB61               956+??0051:  IN      INT1        ;Get Intl status
1363 E602               957+        ANI     BOM         ;Check for byte out
1365 CA6113             958+        JZ      ??0051      ;If not, try again
1368 3E01               959         MVI     A,MODE1     ;Not talk only or listen only
136A D364               960         OUT     ADRMD       ;Enable 91 address mode 1
                        961         CLRA
136C AF                 962+        XRA     A           ;A XOR A =0
136D D365               963         OUT     AUXMD       ;Immediate XEQ PON
136F 3E01               964         MVI     A,MDA       ;My device address
1371 D366               965         OUT     ADR01       ;enabled to talk and listen
1373 3EA1               966         MVI     A,AXRB+CPTEN    ;Command pass thru enable
1375 D365               967         OUT     AUXMD
                        968 ;*******optional PP configuration goes here********
1377 3EF1               969         MVI     A,GIDL      ;92 go idle command
1379 D369               970         OUT     CMD92
                        971         WAITX
137B DB6F               972+??0052:  IN      PRTF
137D E602               973+        ANI     TCIF
137F C27B13             974+        JNZ     ??0052
                        975         WAITT               ;Wait for TCI
1382 DB6F               976+??0053:  IN      PRTF        ;Get task complete int,etc.
1384 E602               977+        ANI     TCIF        ;Mask it
1385 CA8213             978+        JZ      ??0053      ;Wait for task to be complete
1389 23                 979         INX     H
138A C9                 980 PCTL1:  RET
                        981 ;
                        982 ;
                        983 ;*****************************************
                        984 ;
                        985 ;RECEIVE CONTROL ROUTINE
                        986 ;
                        987 ;INPUTS:         None
                        988 ;OUTPUTS:        None
                        989 ;CALLS:          None
                        990 ;DESTROYS:       A, F
                        991 ;RETURNS:        0= invalid (not take control to us or CPT bit not on)
                        992 ;                < > 0 = valid take control-- 92 will now be in control
                        993 ;NOTE:           THIS CODE MUST BE TIGHTLY INTEGRATED INTO ANY USER
                        994 ;                SOFTWARE THAT FUNCTIONS WITH THE 8291 AS  A DEVICE.
                        995 ;                NORMALLY SOME ADVANCE WARNING OF IMPENDING PASS
                        996 ;                CONTROL SHOULD BE GIVEN TO US BY THE CONTROLLER
                        997 ;                WITH OTHER USEFUL INFO. THIS PROTOCOL IS SITUATION
                        998 ;                SPECIFIC AND WILL NOT BE COVERED HERE.
                        999 ;
                       1000 ;
138B DB61               1001 RCTL:  IN      INT1        ;Get INT1 reg (i.e. CPT etc.)
138D E680               1002        ANI     CPT         ;Is command pass thru on ?
138F CACF13             1003        JZ      RCTL2       ;No, invalid-- go return
1392 DB65               1004        IN      CPTRG       ;Get command
1394 FE09               1005        CPI     TCT         ;Is it take control ?
```

231324-43

```
1396 C2CA13      1005          JNZ     RCTL1   ;No, go return invalid
1399 DB64        1007          IN      ADRST   ;Get address status
139B E602        1008          ANI     TA      ;Is TA on ?
139D CACA13      1009          JZ      RCTL1   ;No -- go return invalid
13A0 3E60        1010          MVI     A,DTDL1 ;Disable talker listener
13A2 D365        1011          OUT     ADR01
13A4 3E80        1012          MVI     A,TON   ;Talk only
13A6 D364        1013          OUT     ADRMD
                 1014          CLRA
13A8 AF          1015+         XRA     A       ;A XOR A =0
13A9 D361        1016          OUT     INT1    ;Mask off INT bits
13AB D362        1017          OUT     INT2
13AD D365        1018          OUT     AUXMD
13AF 3EFA        1019          MVI     A,TCNTR ;Take (receive) control 92 command
13B1 D369        1020          OUT     CMD92
13B3 3E0F        1021          MVI     A,VSCMD ;Valid command pattern for 91
13B5 D365        1022          OUT     AUXMD
                 1023 ;******** optional TOUT1 check could be put here ********
                 1024          WAITX
13B7 DB6F        1025+??0054:  IN      PRTF
13B9 E602        1026+         ANI     TCIF
13BB C2B713      1027+         JNZ     ??0054
                 1028          WAITT           ;Wait for TCI
13BE DB6F        1029+??0055:  IN      PRTF    ;Get task complete int,etc.
13C0 E602        1030+         ANI     TCIF    ;Mask it
13C2 CABE13      1031+         JZ      ??0055  ;Wait for task to be complete
13C5 3E09        1032          MVI     A,TCT   ;Valid return pattern
13C7 C3CF13      1033          JMP     RCTL2   ;Only one return per routine
13CA 3E0F        1034 RCTL1:   MVI     A,VSCMD ;Acknowledge CPT
13CC D365        1035          OUT     AUXMD
                 1036          CLRA            ;Error return pattern
13CE AF          1037+         XRA     A       ;A XOR A =0
13CF C9          1038 RCTL2:   RET
                 1039 ;
                 1040 ;*************************************************
                 1041 ;
                 1042 ;        SRQ ROUTINE
                 1043 ;
                 1044 ;INPUTS:         None
                 1045 ;OUTPUTS:        None
                 1046 ;CALLS:          None
                 1047 ;RETURNS:        A= 0 no SRQ
                 1048 ;                A < > 0 SRQ occured
                 1049 ;
                 1050 ;
13D0 DB69        1051 SRQD:    IN      INTST   ;Get 92's INTRQ status
13D2 E620        1052          ANI     SRQBT   ;Mask off SRQ
13D4 CAE213      1053          JZ      SRQD2   ;Not set--- go return
13D7 F608        1054          ORI     IACK    ;Set--- must clear it with IACK
13D9 D369        1055          OUT     CMD92
13DB DB69        1056 SRQD1:   IN      INTST   ;Get IBF
13DD E602        1057          ANI     IBFBT   ;Mask it
13DF CADB13      1058          JZ      SRQD1   ;Wait if not set
13E2 C9          1059 SRQD2:   RET
                 1060 ;
                 1061 ;*****************************************
                 1062 ;
                 1063 ;REMOTE ENABLE ROUTINE
                 1064 ;
                 1065 ;INPUTS:         None
                 1066 ;OUTPUTS:        None
                 1067 ;CALLS:          NONE
                 1068 ;DESTROYS:       A, F
                 1069 ;
13E3 3EF8        1070 REME:    MVI     A,SREM
13E5 D369        1071          OUT     CMD92   ;92 asserts remote enable
                 1072          WAITX           ;Wait for TCI = 0
13E7 DB6F        1073+??0056:  IN      PRTF
13E9 E602        1074+         ANI     TCIF
13EB C2E713      1075+         JNZ     ??0056
                 1076          WAITT           ;Wait for TCI
13EE DB6F        1077+??0057:  IN      PRTF    ;Get task complete int,etc.
13F0 E602        1078+         ANI     TCIF    ;Mask it
13F2 CAEE13      1079+         JZ      ??0057  ;Wait for task to be complete
```

231324-44

```
13F5 C9          1080          RET
                 1081 ;
                 1082 ;*****************************************
                 1083 ;
                 1084 ;LOCAL ROUTINE
                 1085 ;
                 1086 ;
                 1087 ;INPUTS:          None
                 1088 ;OUTPUTS:         None
                 1089 ;CALLS:           None
                 1090 ;DESTROYS:        A, F
                 1091 ;
13F6 3EF7        1092 LOCL:    MVI     A,SLOC
13F8 D369        1093          OUT     CMD92    ;92 stops asserting remote enable
                 1094          WAITX            ;Wait for TCI =0
13FA DB6F        1095+??0058:  IN      PRTF
13FC E602        1096+         ANI     TCIF
13FE C2FA13      1097+         JNZ     ??0058
                 1098          WAITT            ;Wait for TCI
1401 DB6F        1099+??0059:  IN      PRTF     ;Get task complete int,etc.
1403 E602        1100+         ANI     TCIF     ;Mask it
1405 CA0114      1101+         JZ      ??0059   ;Wait for task to be complete
1408 C9          1102          RET
                 1103 ;
                 1104 ;*********************************************
                 1105 ;
                 1106 ;INTERFACE CLEAR / ABORT ROUTINE
                 1107 ;
                 1108 ;
                 1109 ;INPUTS:              None
                 1110 ;OUTPUTS:             None
                 1111 ;CALLS:               None
                 1112 ;DESTROYS:            A, F
                 1113 ;
                 1114 ;
1409 3EF9        1115 IFCL:    MVI     A,ABORT
140B D369        1116          OUT     CMD92    ;Send IFC
                 1117          WAITX            ;Wait for TCI =0
140D DB6F        1118+??0060:  IN      PRTF
140F E602        1119+         ANI     TCIF
1411 C20D14      1120+         JNZ     ??0060
                 1121          WAITT            ;Wait for TCI
1414 DB6F        1122+??0061:  IN      PRTF     ;Get task complete int,etc.
1416 E602        1123+         ANI     TCIF     ;Mask it
1418 CA1414      1124+         JZ      ??0061   ;Wait for task to be complete
                 1125 ;Delete both WAITX & WAITT if this routine
                 1126 ;is to be called while the 9292 is
                 1127 ;Controller-in-Charge. If not C.I.C. then
                 1128 ;TCI is set, else nothing is set (IFC is sent)
                 1129 ;and the WAIT'S will hang forever
141B C9          1130          RET
                 1132 ;
```

231324-45

3

```
                    1133 ;APPLICATION EXAMPLE CODE FOR 8085
                    1134 ;
0032                1135 FGDNL   EQU     '2'         ;Func gen device num "2" ASCII,lstn
0031                1136 FCDNL   EQU     '1'         ;Freq ctr device num "1" ASCII,lstn
0051                1137 FCDNT   EQU     'Q'         ;Freq ctr talk address
000D                1138 CR      EQU     0DH         ;ASCII carriage return
000A                1139 LF      EQU     0AH         ;ASCII line feed
00FF                1140 LEND    EQU     0FFH        ;List end for Talk/Listen lists
0040                1141 SRQM    EQU     40H         ;Bit indicating device sent SRQ
                    1142 ;
141C 46553146       1143 FGDATA: DB      'FU1FR37KHAM2VO',CR     ;Data to set up func. gen
1420 5233374B
1424 48414D32
1428 564F
142A 0D
000F                1144 LIM1    EQU     15                  ;Buffer length
1428 50463447       1145 FCDATA: DB      'PF4G7T'            ;Data to set up freq ctr
142F 3754
0006                1146 LIM2    EQU     6                   ;Buffer length
1431 31             1147 LL1:    DB      FCDNL,LEND          ;Listen list for freq ctr
1432 FF
1433 32             1148 LL2:    DB      FGDNL,LEND          ;Listen list for func. gen
1434 FF
1435 51             1149 TL1:    DB      FCDNT,LEND          ;Talk list for freq ctr
1436 FF
                    1150 ;
                    1151 ;SETUP FUNCTION GENERATOR
1437 060D           1152         MVI     B,CR    ;EOS
1439 0E0F           1153         MVI     C,LIM1  ;Count
143B 111C14         1154         LXI     D,FGDATA        ;Data pointer
143E 213314         1155         LXI     H,LL2   ;Listen list pointer
1441 CD1C10         1156         CALL    SEND
                    1157 ;
                    1158 ;SETUP FREQ COUNTER
                    1159 ;
1444 0654           1160         MVI     B,'T'   ;EOS
1446 0E06           1161         MVI     C,LIM2  ;Count
1448 112A14         1162         LXI     D,FCDATA        ;Data pointer
144B 213114         1163         LXI     H,LL1   ;Listen list pointer
144E CD1C10         1164         CALL    SEND
                    1165 ;
                    1166 ;WAIT FOR SRQ FROM FREQ CTR
                    1167 ;
1451 CDD013         1168 LOOP:   CALL    SRQD    ;Has SRQ occurred ?
1454 CA5114         1169         JZ      LOOP    ;No, wait for it
                    1170 ;
                    1171 ;SERIAL POLL TO CLEAR SRQ
                    1172 ;
1457 11003C         1173         LXI     D,SPBYTE        ;Buffer pointer
145A 213514         1174         LXI     H,TL1           ;Talk list pointer
145D CD1C12         1175         CALL    SPOL
1460 1B             1176         DCX     D       ;Backup buffer pointer to ctr byte
1461 1A             1177         LDAX    D       ;Get status byte
1462 E640           1178         ANI     SRQM    ;Did ctr assert SRQ ?
1464 CA7714         1179         JZ      ERROR   ;Ctr should have said yes
                    1180 ;
                    1181 ;RECEIVE READING FROM COUNTER
                    1182 ;
1467 060A           1183         MVI     B,LF    ;EOS
1469 0E11           1184         MVI     C,LIM3  ;Count
146B 213514         1185         LXI     H,TL1   ;Talk list pointer
146E 11013C         1186         LXI     D,FCDATI        ;Data in buffer pointer
1471 CD9F10         1187         CALL    RECV
1474 C27714         1188         JNZ     ERROR
                    1189 ;
                    1190 ;******* rest of user processing goes here *****
                    1191 ;
                    1192 ;
1477 00             1193 ERROR:  NOP             ;User dependant error handling
                    1194 ;       ETC.
3C00                1195 ORG     3C00H
3C00                1196 SPBYTE: DS      1       ;Location for serial poll byte
0011                1197 LIM3    EQU     17      ;Max freq counter input
```

231324-46

```
3C01                1198 FCDATI: DS      LIM3    ;Freq ctr input buffer
                    1199         END

PUBLIC SYMBOLS


EXTERNAL SYMBOLS

USER SYMBOLS
ABORT  A 00F9   ADR01  A 0056   ADRMD  A 0064   ADRST  A 0054   AUXMD  A 0055   AXRA   A 0080   AXRB   A 00A0
BIM    A 0001   BOF    A 0001   BOM    A 0002   BUSST  A 0058   CAHCY  A 0003   CLKRT  A 0023   CLRA   + 0007
CLRST  A 0069   CMD92  A 0069   CPT    A 0089   CPTEN  A 0001   CPTRG  A 0055   CR     A 000D   DCL    A 0014
DCLR   A 11EC   DCLR1  A 11F0   DCLR2  A 1209   DIN    A 0059   DOUT   A 0050   DTDL1  A 0050   DTDL2  A 00E0
EDEOS  A 0004   ENDMK  A 0019   EOIS   A 0008   EOIST  A 0020   EOSR   A 0057   ERFLG  A 0068   ERR4   A 0068
ERROR  A 1477   EVBIT  A 0010   EVCST  A 0068   EVREG  A 0058   EXPP   A 00F5   FCDATA A 1428   FCDATI A 3C01
FCONL  A 0031   FCONT  A 0051   FGDATA A 141C   FGONL  A 0032   FNHSK  A 0003   GET    A 0008   GIDL   A 00F1
GSEC   A 00F4   GTS8   A 00F6   HOEND  A 0002   HOHSK  A 0001   IACK   A 003B   IBFBT  A 0002   IBFF   A 0010
IFCL   A 1409   INIT   A 1000   INT1   A 0061   INT2   A 0052   INTM   A 03A0   INTM1  A 0351   INTMR  A 0368
INTST  A 0059   LA     A 0001   LEND   A 00FF   LF     A 000A   LIM1   A 000F   LIM2   A 0006   LIM3   A 0011
LL1    A 1431   LL2    A 1433   LOCL   A 13F5   LON    A 0040   LOOP   A 1451   MDA    A 0001   MLA    A 0021
MODE1  A 0001   MTA    A 0041   NVCMD  A 0007   OBFF   A 0038   PCTL   A 1344   PCTL1  A 138A   PPC    A 0005
PPD    A 0070   PPDS   A 12E0   PPDS1  A 12E4   PPDS2  A 12FD   PPE    A 0059   PPEN   A 12A3   PPEN1  A 12A7
PPEN2  A 12D8   PPOL   A 1327   PPU    A 0015   PPUN   A 131B   PRT91  A 0050   PRT92  A 0058   PRTF   A 005F
RANGE  + 0005   RBST   A 00E7   RCST   A 00E6   RCTL   A 1398   RCTL1  A 13CA   RCTL2  A 13CF   RECV   A 109F
RECV1  A 10EA   RECV2  A 1105   RECV3  A 1106   RECV4  A 1110   RECV5  A 1117   RECV6  A 1139   REME   A 13E3
RERF   A 00E4   RERM   A 00EA   REVC   A 00E3   RINM   A 00E5   RSET   A 00F2   RSTI   A 00F3   RTOUT  A 00E9
SDEOI  A 0006   SEND   A 101C   SEND1  A 102E   SEND2  A 1047   SEND3  A 1069   SEND4  A 1070   SEND5  A 107F
SEND6  A 1088   SETF   + 0003   SLOC   A 00F7   SPBYTE A 3C00   SPCNI  A 00F0   SPD    A 0019   SPE    A 0018
SPIF   A 0004   SPOL   A 121C   SPOL1  A 123D   SPOL2  A 1294   SREM   A 00F8   SRQBT  A 0020   SRQD   A 13D0
SRQD1  A 13DB   SRQD2  A 13E2   SROM   A 0040   STCNI  A 00FE   TA     A 0002   TCASY  A 00FC   TCIF   A 0002
TCNTR  A 00FA   TCSY   A 00FD   TCT    A 0009   TL1    A 1435   TLON   A 00C0   TON    A 0098   TOREG  A 0068
TOST   A 0068   TOUT1  A 0001   TOUT2  A 0002   TOUT3  A 0004   TRIG   A 11BC   TRIG1  A 11C0   TRIG2  A 11D9
UNL    A 003F   VSCMD  A 000F   WAITI  + 0002   WAITO  + 0001   WAITT  + 0004   WAITX  + 0003   WEVC   A 00E2
WOUT   A 0021   XFER   A 113A   XFER1  A 1153   XFER2  A 116C   XFER3  A 1193   XFER4  A 11BB

ASSEMBLY COMPLETE,  NO ERRORS
```

231324-47

3

APPENDIX B

Test Cases for the Software Drivers

The following test cases were used to exercise the software routines and to check their action. To provide another device/controller on the GPIB a ZT488 GPIB Analyzer was used. This analyzer acted as a talker, listener or another controller as needed to execute the tests. The sequence of outputs are shown with each test. All numbers are hexadecimal.

Send Test Cases

B=	44	44	44
C=	30	2	0
DE=	3E80	3E80	3E80
HL=	3E70	3E70	3E70
3E70:	20 30 3E 3F		
3E80:	11 44		
GPIB output:	41 ATN	41 ATN	41 ATN
	3F ATN	3F ATN	3F ATN
	20 ATN	20 ATN	20 ATN
	30 ATN	30 ATN	30 ATN
	3E ATN	3E ATN	3E ATN
	11	11	
	44 EOI	44 EOI	
Ending B=	44	44	44
Ending C=	2E	0	0
Ending DE=	3E82	3E82	3E80
Ending HL=	3E73	3E73	3E73

Receive Test Cases

B=	44	44	44	44	44	44	44
C=	30	30	30	30	4	4	0=256
DE=	3E80	3E80	3E80	3E80	3E80	3E80	3E80
HL=	3E70	3E70	3E70	3E70	3E70	3E70	3E70
3E70:	40	50	5E	5F	40	40	40
GPIB output:	40 ATN	50 ATN	5E ATN		40 ATN	40 ATN	40 ATN
	3F ATN	3F ATN	3F ATN		3F ATN	3F ATN	3F ATN
	21 ATN	21 ATN	21 ATN		21 ATN	21 ATN	21 ATN
ZT488 Data	1	1	1		1	11	1
In	2	2	2		2	22	2
	3	3	3		3	33	3
	4	4	44,EOI		4	44	44
	44	5,EOI					
Ending A =	0	0	0	5F	40	0	0
Ending B =	0	0	0	44	40	0	0
Ending C =	2B	2B	2C	30	0	0	FC
Ending DE=	3E85	3E85	3E84	3E80	3E84	3E84	3E84
Ending HL=	3E71	3E71	3E71	3E70	3E71	3E71	3E71

Serial Poll Test Cases

C=	30	C=	30
DE=	3E80	DE=	3E80
HL=	3E70	HL=	3E70
3E70:	40	3E70:	5F
	50	GPIB output:	3F ATN
	5E		21 ATN
	5F		18 ATN
GPIB output:	3F ATN		19 ATN
output:	21 ATN	Ending C =	30
output:	18 ATN	Ending DE=	3E80
output:	40 ATN	Ending HL=	3E70
input*:	00		
output:	50 ATN		
input*:	41		
output:	5E ATN		
input*:	7F		
output:	19 ATN		

*NOTE: leave ZT488 in single step mode even on input

Ending C =	30
Ending DE=	3E83
Ending HL=	3E73
Ending 3E80:	00 41 7F

Pass Control Test Cases

HL=	3E70	3E70	3E70
3E70:	40	41(MTA)	5F
GPIB output:	40 ATN		
	09 ATN		
	—$\overline{\text{ATN}}$		
Ending HL=	3E71	3E70	3E70
Ending A =	02	41(MTA)	5F

Receive Control Test Cases

GPIB input	10 ATN	40 ATN	41 ATN
	$\overline{\text{ATN}}$	09 ATN	09 ATN
Run Receive Control			
GPIB Input		$\overline{\text{ATN}}$	$\overline{\text{ATN}}$
Ending A=	0	0	09

	3E ATN	
	05 ATN	
	63 ATN	
Ending DE =	3E83	3E80
Ending HL =	3E73	3E70

Parallel Poll Disable Test Cases

HL =	3E70	3E70
3E70:	20 30 3E 3F	3F
GPIB output:	3F ATN	3F ATN
	20 ATN	05 ATN
	30 ATN	70 ATN
	3E ATN	
	05 ATN	
	70 ATN	
Ending HL =	3E73	3E70

Parallel Poll Unconfigure Test Case

GPIB output: 15 ATN

Parallel Poll Test Cases

Set DIO #	1	2	3	4	5	6	7	8	None
Ending A	1	2	4	8	10	20	40	80	0

SRQ Test

	Set SRQ momentarily	Reset SRQ
Ending A =	02	00

Trigger Test

```
HL=       3E70
DE=       3E80
BC=       4430
3E70:     20  30  3E  3F
GPIB output:  3F ATN
              20 ATN
              30 ATN
              3E ATN
              08 ATN
Ending HL=    3E73
DE=           3E80
BC=           4430
```

Device Clear Test

```
HL=           3E70
DE=           3E80
BC=           4430
3E70:         20  30  3E  3F
GPIB output:  3F ATN
              20 ATN
              30 ATN
              3E ATN
              14 ATN
Ending HL=    3E73
DE=           3E80
RC=           4430
```

XFER Test

```
B=            44
HL=           3E70:
3E70:         40  20  30  3E  3F
GPIB output:  40 ATN
              3F ATN
              20 ATN
              30 ATN
              3E ATN
GPIB input:   0
              1
              2
              3
              44
Ending A =    0
B=            44
HL =          3E74
```

Application Example
GPIB Output/Input

GPIB output:	41 ATN
	3F ATN
	32 ATN
	46
	55
	31
	46
	52
	33
	37
	4B
	48
	41
	4D
	32
	56
	4F
	0D EOI
	41 ATN
	3F ATN
	31 ATN
	50
	46
	34
	47
	37
	54 EOI
GPIB input:	SRQ
GPIB output:	3F ATN
	21 ATN
	18 ATN
	51 ATN
GPIB input:	40 $\overline{\text{SRQ}}$
GPIB output:	19 ATN
	51 ATN
	3F ATN
	21 ATN

GPIB input:

20
2B
20
20
20
33
37
30
30
30
2E
30
45
2B
30
0D
0A

GPIB output: XX ATN

APPENDIX C

REMOTE MESSAGE CODING

Mnemonic	Message Name		Type	Class	DIO8	7	6	5	4	3	2	1	NN DRD AFA VDC	DIO ATN	EOI	SRQ	IFC	REN
			Bus Signal Line(s) and Coding That Asserts the True Value of the Message															
ACG	addressed command group		M	AC	Y	0	0	0	X	X	X	X	XXX	1	X	X	X	X
ATN	attention		U	UC	X	X	X	X	X	X	X	X	XXX	1	X	X	X	X
DAB	data byte	(Notes 1, 9)	M	DD	D8	D7	D6	D5	D4	D3	D2	D1	XXX	0	X	X	X	X
DAC	data accepted		U	HS	X	X	X	X	X	X	X	X	XX0	X	X	X	X	X
DAV	data valid		U	HS	X	X	X	X	X	X	X	X	1XX	X	X	X	X	X
DCL	device clear		M	UC	Y	0	0	1	0	1	0	0	XXX	1	X	X	X	X
END	end		U	ST	X	X	X	X	X	X	X	X	XXX	0	1	X	X	X
EOS	end of string	(Notes 2, 9)	M	DD	E8	E7	E6	E5	E4	E3	E2	E1	XXX	0	X	X	X	X
GET	group execute trigger		M	AC	Y	0	0	0	1	0	0	0	XXX	1	X	X	X	X
GTL	go to local		M	AC	Y	0	0	0	0	0	0	1	XXX	1	X	X	X	X
IDY	identify		U	UC	X	X	X	X	X	X	X	X	XXX	X	1	X	X	X
IFC	interface clear		U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	1	X
LAG	listen address group		M	AD	Y	0	1	X	X	X	X	X	XXX	1	X	X	X	X
LLO	local lock out		M	UC	Y	0	0	1	0	0	0	1	XXX	1	X	X	X	X
MLA	my listen address	(Note 3)	M	AD	Y	0	1	L5	L4	L3	L2	L1	XXX	1	X	X	X	X
MTA	my talk address	(Note 4)	M	AD	Y	1	0	T5	T4	T3	T2	T1	XXX	1	X	X	X	X
MSA	my secondary address	(Note 5)	M	SE	Y	1	1	S5	S4	S3	S2	S1	XXX	1	X	X	X	X
NUL	null byte		M	DD	0	0	0	0	0	0	0	0	XXX	X	X	X	X	X
OSA	other secondary address		M	SE	(OSA = SCG ∧ $\overline{\text{MSA}}$)													
OTA	other talk address		M	AD	(OTA = TAG ∧ $\overline{\text{MTA}}$)													
PCG	primary command group		M	—	(PCG = ACG ∨ UCG ∨ LAG ∨ TAG)													
PPC	parallel poll configure		M	AC	Y	0	0	0	0	1	0	1	XXX	1	X	X	X	X
PPE	parallel poll enable	(Note 6)	M	SE	Y	1	1	0	S	P3	P2	P1	XXX	1	X	X	X	X
PPD	parallel poll disable	(Note 7)	M	SE	Y	1	1	1	D4	D3	D2	D1	XXX	1	X	X	X	X
PPR1	parallel poll response 1	(Note 10)	U	ST	X	X	X	X	X	X	X	1	XXX	1	1	X	X	X
PPR2	parallel poll response 2	(Note 10)	U	ST	X	X	X	X	X	X	1	X	XXX	1	1	X	X	X

REMOTE MESSAGE CODING (Continued)

Mnemonic	Message Name		Type	Class	Bus Signal Line(s) and Coding That Asserts the True Value of the Message: DIO8	7	6	5	4	3	2	DIO1	NNDRDAFAVDC	ATN	EOI	SRQ	IFC	REN
PPR3	parallel poll response 3	(Note 10)	U	ST	X	X	X	X	X	1	X	X	XXX	1	1	X	X	X
PPR4	parallel poll response 4		U	ST	X	X	X	X	1	X	X	X	XXX	1	1	X	X	X
PPR5	parallel poll response 5		U	ST	X	X	X	1	X	X	X	X	XXX	1	1	X	X	X
PPR6	parallel poll response 6	(Note 10)	U	ST	X	X	1	X	X	X	X	X	XXX	1	1	X	X	X
PPR7	parallel poll response 7		U	ST	X	1	X	X	X	X	X	X	XXX	1	1	X	X	X
PPR8	parallel poll response 8		U	ST	1	X	X	X	X	X	X	X	XXX	1	1	X	X	X
PPU	parallel poll unconfigure		M	UC	Y	0	0	1	0	1	0	1	XXX	1	X	X	X	X
REN	remote enable		U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	X	1
RFD	ready for data		U	HS	X	X	X	X	X	X	X	X	X0X	X	X	X	X	X
RQS	request service	(Note 9)	U	ST	X	1	X	X	X	X	X	X	XXX	0	X	X	X	X
SCG	secondary command group		M	SE	Y	1	1	X	X	X	X	X	XXX	1	X	X	X	X
SDC	selected device clear		M	AC	Y	0	0	0	0	1	0	0	XXX	1	X	X	X	X
SPD	serial poll disable		M	UC	Y	0	0	1	1	0	0	1	XXX	1	X	X	X	X
SPE	serial poll enable		M	UC	Y	0	0	1	1	0	0	0	XXX	1	X	X	X	X
SRQ	service request		U	ST	X	X	X	X	X	X	X	X	XXX	X	X	1	X	X
STB	status byte	(Notes 8, 9)	M	ST	S8	X	S6	S5	S4	S3	S2	S1	XXX	0	X	X	X	X
TCT	take control		M	AC	Y	0	0	0	1	0	0	1	XXX	1	X	X	X	X
TAG	talk address group		M	AD	Y	1	0	X	X	X	X	X	XXX	1	X	X	X	X
UCG	universal command group		M	UC	Y	0	0	1	X	X	X	X	XXX	1	X	X	X	X
UNL	unlisten		M	AD	Y	0	1	1	1	1	1	1	XXX	1	X	X	X	X
UNT	untalk	(Note 11)	M	AD	Y	1	0	1	1	1	1	1	XXX	1	X	X	X	X

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:

1. D1–D8 specify the device dependent data bits.
2. E1–E8 specify the device dependent code used to indicate the EOS message.
3. L1–L5 specify the device dependent bits of the device's listen address.
4. T1–T5 specify the device dependent bits of the device's talk address.
5. S1–S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.

S	Response
0	0
1	1

P1–P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

7. D1–D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1–S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use, see 6.3.

U	SI	X	X	1	X	X	X	X	X	XXX	1	1	X	X	X	X	X	X	X
U	ST	X	1	X	X	X	X	X	X	XXX	1	1	X	X	X	X	X	X	X
U	ST	1	X	X	X	X	X	X	X	XXX	1	1	X	X	X	X	X	X	X
M	UC	Y	0	0	1	0	1	0	1	XXX	1	X	X	X	X	X	X	X	X
U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	X	X	X	X	X	1
U	HS	X	X	X	X	X	X	X	X	X0X	X	X	X	X	X	X	X	X	X
U	ST	X	1	X	X	X	X	X	X	XXX	0	X	X	X	X	X	X	X	X
M	SE	Y	1	1	X	X	X	X	X	XXX	1	X	X	X	X	X	X	X	X
M	AC	Y	0	0	0	0	1	0	0	XXX	1	X	X	X	X	X	X	X	X
M	UC	Y	0	0	1	1	0	0	1	XXX	1	X	X	X	X	X	X	X	X
M	UC	Y	0	0	1	1	0	0	0	XXX	1	X	X	X	X	X	X	X	X
U	ST	X	X	X	X	X	X	X	X	XXX	X	X	X	1	1	X	X	X	X
M	ST	S	X	S	S	S	S	S	S	XXX	0	X	X	X	X	X	X	X	X
				8		6	5	4	3	2	1								
M	AC	Y	0	0	0	1	0	0	1	XXX	1	X	X	X	X	X	X	X	X
M	AD	Y	1	0	X	X	X	X	X	XXX	1	X	X	X	X	X	X	X	X
M	UC	Y	0	0	1	X	X	X	X	XXX	1	X	X	X	X	X	X	X	X
M	AD	Y	0	1	1	1	1	1	1	XXX	1	X	X	X	X	X	X	X	X
M	AD	Y	1	0	1	1	1	1	1	XXX	1	X	X	X	X	X	X	X	X

The 110 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

NOTES:

1. D1–D8 specify the device dependent data bits.
2. E1–E8 specify the device dependent code used to indicate the EOS message.
3. L1–L5 specify the device dependent bits of the device's listen address.
4. T1–T5 specify the device dependent bits of the device's talk address.
5. S1–S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.

S	Response
0	0
1	1

P1–P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

7. D1–D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1–S8, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use, see 8.3.

Modem Products

4

4 Modern Products

89024
2400 BPS INTELLIGENT MODEM CHIP SET

- For Public Switched Telephone Network and Unconditioned Leased Line Applications
- V.22 bis, V.22 A/B, V.21, Bell 212A, and Bell 103 Compatible
- Serial Command Set Compatible with Hayes* Smartmodem 2400*
- Automatically Adapts to Remote Modem Type with Recognition of Data Rates
- DTMF and Pulse Dialing
- On-Chip Hybrid and Billing Delay Timer
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Telephone Line Audio Monitor Output
- Analog/Digital Loopback Diagnostics with Mark/Space Pattern Generation and Error Detection
- Simple Serial Interface to External NVRAM
- Easily Customized Command Set and Features
- Two Chip Intelligent Modem Solution with Minimal External Components
- Output Level Programmable over 16 dB Range
- Dial and Re-dial Capability
- Full Set of Control Signals for DAA Interface
- Local, External, or Slave Timing Options in Synchronous Mode
- Adaptive Equalization
- Capable of Detecting Dial, Busy, Ringback and Modem Answer Tones of Most International Networks
- Auxiliary Relay Control Output

4

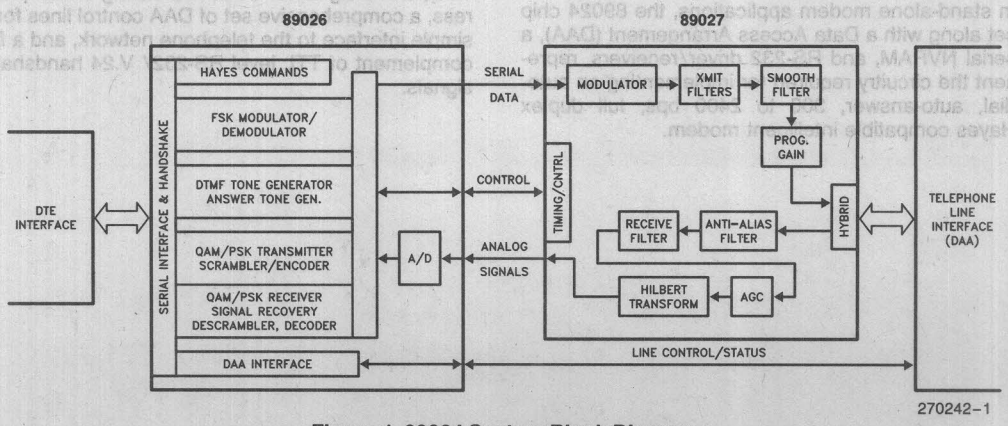


Figure 1. 89024 System Block Diagram

*Hayes, Smartmodem 2400 and Smartcom II are registered trademarks of Hayes Microcomputer Products, Inc.

September 1990
Order Number: 270242-005

GENERAL DESCRIPTION

The Intel 89024 chip set is a highly integrated, high performance, intelligent modem, providing a complete system in two chips. The system is compatible with the following CCITT and Bell standards:

- CCITT V.22 bis
  2400 bps sync and async
  1200 bps sync and async (fall-back)
- CCITT V.22 A & B
  1200 bps sync and async
- CCITT V.21
  0 to 300 bps anisochronous
- BELL 212A
  1200 bps sync and async
  300 bps fall-back mode
- BELL 103
  0 to 300 bps anisochronous

The 89024 system consists of a 16 bit application specific processor (89026) and an analog front end device (89027). The 89026 processor performs all "Digital Signal Processing" algorithm execution for processing the modem signals, as well as providing all modem control functions typically performed by an external processor. The analog front end provides for 2 wire and 4 wire telephone line interface, D/A conversion, and most of the complex filtering functions required in QAM/PSK/FSK modems. Refer to Figure 1 for a simplified block diagram of the system.

In stand-alone modem applications, the 89024 chip set along with a Data Access Arrangement (DAA), a serial NVRAM, and RS-232 driver/receivers, represent the circuitry required for implementing an auto-dial, auto-answer, 300 to 2400 bps, full duplex Hayes compatible intelligent modem.

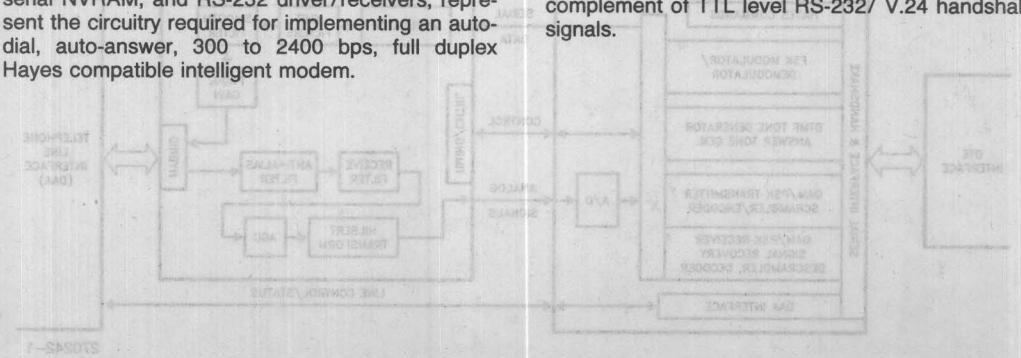
A complete set of industry standard AT commands is provided for modem configuration and user interface. Virtually all PC software written for the Hayes Smartmodem 2400 can also be used with this chip set. Alternatively, in applications where user proprietary modem control commands and features are desired, the user can replace the 89024 internal command module with custom proprietary software resident in the 89026 microcontroller's on-chip ROM or an external memory device.

The 89024 supports two versions of firmware. These are internal, which is asynchronous only and external, which is asynchronous and synchronous. Any differences in operation are highlighted with footnotes.

The 89024 has a set of default features. Upon power up, the modem configuration will be in accordance with these default options, unless a different configuration has been saved in the external NVRAM with the &W command.

The 89024 modem has built in auto-dialing and auto-answering capabilities. It can be configured to the proper line signaling mode (Tone or Pulse), and to to the type (CCITT or Bell) and speed of the calling or answering modem. It can also detect and identify call set-up signals of telephone networks, allowing unattended data call operation.

A full set of diagnostic loop-test features compatible with CCITT V.54 is supported. The chip set also provides a line signal for audio monitoring of call progress, a comprehensive set of DAA control lines for a simple interface to the telephone network, and a full complement of TTL level RS-232/ V.24 handshake signals.



PACKAGING

The 89027 is available in PLCC and standard plastic DIP packages. The 89026 is available in a PLCC package.

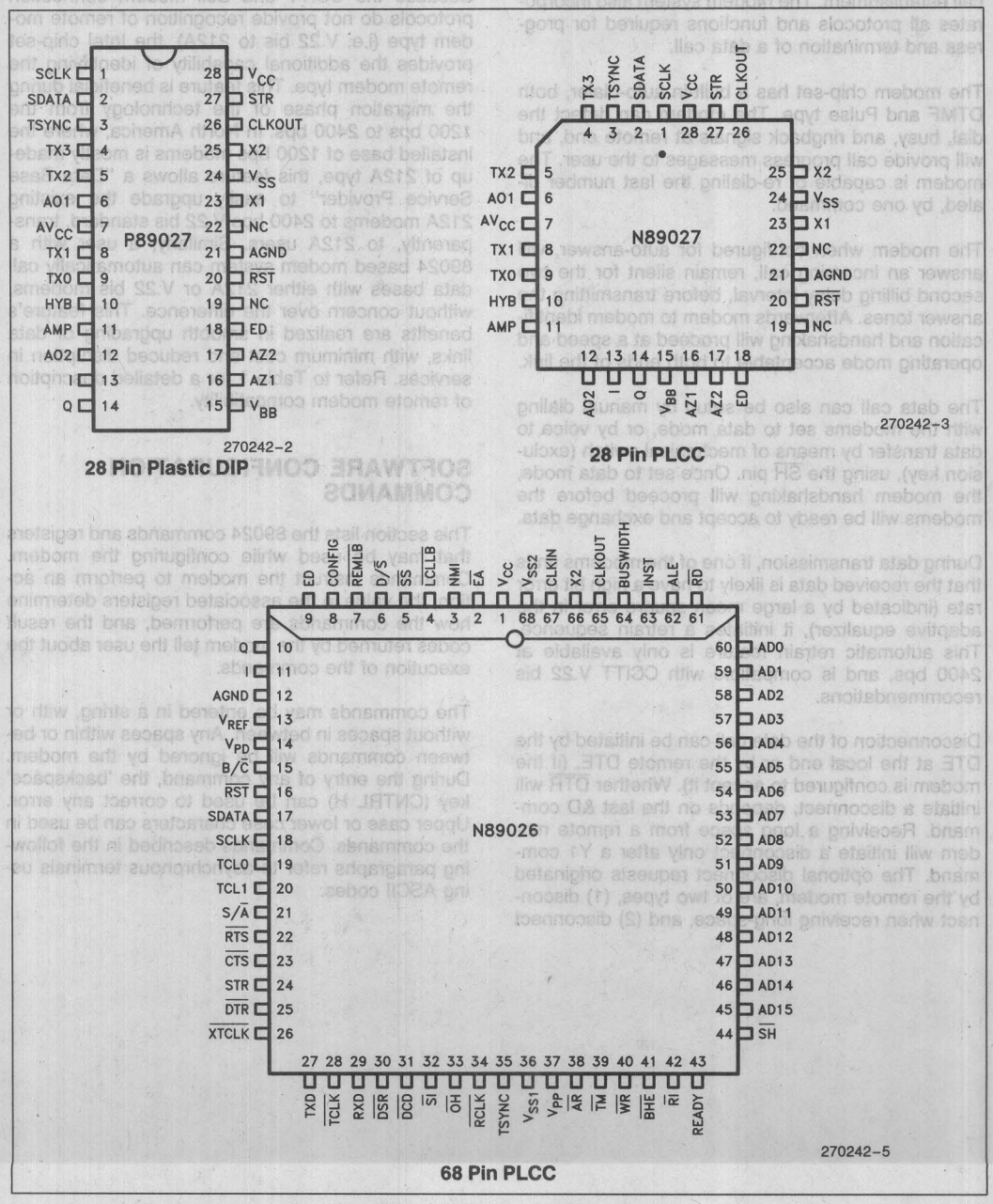


Figure 2. Device Packages

CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89024 modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip-set has a built-in auto-dialer, both DTMF and Pulse type. The modem can detect the dial, busy, and ringback signals at remote end, and will provide call progress messages to the user. The modem is capable of re-dialing the last number dialed, by one command.

The modem when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, before transmitting the answer tones. Afterwards modem to modem identification and handshaking will proceed at a speed and operating mode acceptable to both ends of the link.

The data call can also be setup by manual dialing with the modems set to data mode, or by voice to data transfer by means of mechanical switch (exclusion key), using the $\overline{SH}$ pin. Once set to data mode, the modem handshaking will proceed before the modems will be ready to accept and exchange data.

During data transmission, if one of the modems finds that the received data is likely to have a high bit error rate (indicated by a large mean square error in the adaptive equalizer), it initiates a retrain sequence. This automatic retrain feature is only available at 2400 bps, and is compatible with CCITT V.22 bis recommendations.

Disconnection of the data call can be initiated by the DTE at the local end or by the remote DTE, (if the modem is configured to accept it). Whether $\overline{DTR}$ will initiate a disconnect, depends on the last &D command. Receiving a long space from a remote modem will initiate a disconnect only after a Y1 command. The optional disconnect requests originated by the remote modem, are of two types, (1) disconnect when receiving long-space, and (2) disconnect when received carrier is dropped. The modem chip-set can also be configured to transmit 'long-space' just before disconnection, in each of the aforementioned cases.

Because the CCITT and Bell modem connection protocols do not provide recognition of remote modem type (i.e. V.22 bis to 212A), the Intel chip-set provides the additional capability of identifying the remote modem type. This feature is beneficial during the migration phase of the technology from the 1200 bps to 2400 bps. In North America, where the installed base of 1200 bps modems is mostly made-up of 212A type, this feature allows a "Data Base Service Provider" to easily upgrade the existing 212A modems to 2400 bps V.22 bis standard, transparently, to 212A users. Similarly, a user with a 89024 based modem system can automatically call data bases with either 212A or V.22 bis modems, without concern over the difference. This feature's benefits are realized in smooth upgrading of data links, with minimum cost and reduced disruption in services. Refer to Table 1 for a detailed description of remote modem compatibility.

SOFTWARE CONFIGURATION COMMANDS

This section lists the 89024 commands and registers that may be used while configuring the modem. Commands instruct the modem to perform an action, the value in the associated registers determine how the commands are performed, and the result codes returned by the modem tell the user about the execution of the commands.

The commands may be entered in a string, with or without spaces in between. Any spaces within or between commands will be ignored by the modem. During the entry of any command, the 'backspace' key (CNTRL H) can be used to correct any error. Upper case or lower case characters can be used in the commands. Commands described in the following paragraphs refer to asynchronous terminals using ASCII codes.

Table 1. Remote Modem Compatibility

Originating 89024 Modem		Answering Modem Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	300	—	300*	300*
	1200	1200*	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	1200*	1200	—	1200	1200
	2400	1200*	1200		1200	2400

Answering 89024 Modem		Originating Modem Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	1200	—	1200	1200
	1200	300	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	300*	1200	—	1200	1200
	2400	300*	1200	—	1200	2400

* These connection data rates are obtained when connecting 89024 based modems end to end. The same results may not be obtained when a 89024 based modem is connected to other modems.

Command Set

AT	Attention code.
A	Go off-hook in answer mode
A/	Repeat previous command string
Bn(1)	BELL/CCITT Protocol Compatibility at 1200 bps
Ds	The dialing commands (0-9 A B C D * # P R T S W , ; @)
En	Echo command (En)
Hn	Switch-Hook Control
	If &J1 option is selected, H1 will also switch the auxiliary relay
In	Request Product Code and Checksum
Ln	Speaker Volume
Mn	Monitor On/Off
O	On-Line
Qn	Result Codes
Sn=x	Write S Register
Sn?	Read S Register
Vn	Enable Short-Form Result Codes
Xn	Enable Extended Result Code
Yn	Enable Long Space Disconnect
Z	Fetch Configuration Profile
+++	The Default Escape Code

& Command Set

&C	DCD Options
&D	DTR Options
&F	Fetch Factory Configuration Profile
&G	Guard Tone
&J	Telephone Jack Selection
&L	Leased/Dial-up Line Selection
&M(1)	Async/Sync Mode Selection
&P	Make/Break Pulse Ratio
&R	RTS/CTS Options
&S	DSR Options
&T	Test Commands
&W	Write Configuration to Non Volatile Memory
&X(1)	Sync Clock Source
&Z	Store Telephone Number

NOTE:
1. Available in external code only.

4

CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

S0*	Ring to Answer
S1	Ring Count. (Read Only)
S2	Escape Code Character
S3	Carriage Return Character
S4	Line Feed Character
S5	Back Space Character
S6	Wait for Dial Tone
S7	Wait for Data Carrier
S8	Pause Time for the Comma Dial Modifier
S9	Carrier Detect Response Time
S10	Lost Carrier to Hang Up Delay
S11**	DTMF Tone Duration
S12	Escape Code Guard Time
S13	Not Used
S14 *	Bit Mapped Option Register
S15	Not Used
S16	Modem Test Options
S17	Not Used
S18 *	Test Timer
S19	Not Used
S20	Not Used
S21 *	Bit Mapped Options Register
S22 *	Bit Mapped Options Register
S23 *	Bit Mapped Options Register
S24	Not Used
S25 *	Delay to DTR (Sync Only)
S26 *	RTS to CTS Delay (Half Dup.)
S27 *	Bit Mapped Options Register

NOTE:
* These S registers can be stored in the NVRAM.
**Available in internal code only.

Dial Modifiers

P	Pulse Dial
R	Originate call in Answer Mode
T	Tone Dial
S	Dial a stored number
W	Wait for dial tone
,	Delay a dial sequence
;	Return to command state
!	Initiate a flash
@	Wait for quiet

Example:

Terminal: AT &Z T 1 (602) 555-1212
Modem: OK

Result: Modem stores T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:

Terminal: AT DS
Modem: T16025551212

or by turning on $\overline{\text{DTR}}$ when in Synchronous Mode 2. Up to 33 symbols (dial digits and dial modifiers) may be stored. Spaces and other delimiters are ignored and do not need to be included in the count. If more than 33 symbols are supplied, the dial string will be truncated to 33.

APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 3. The DAA section shown in this diagram may be obtained with FCC registration, or implemented using the suggested diagram in Figure 4.

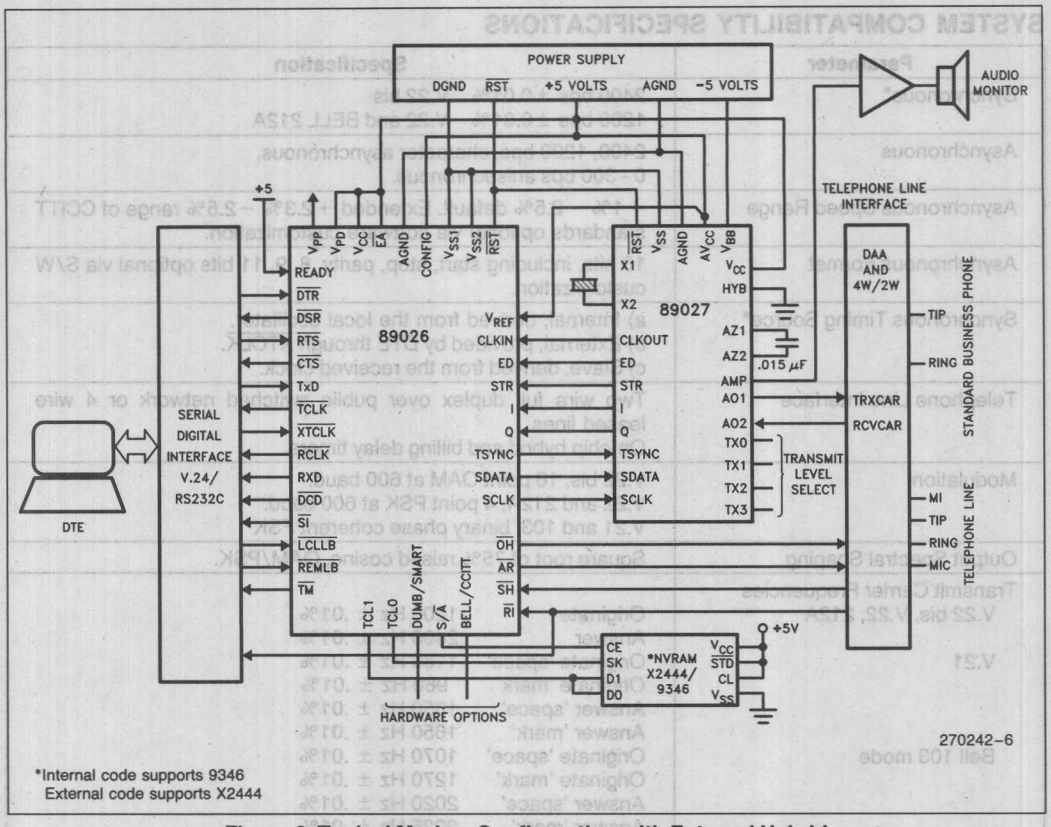


Figure 3. Typical Modem Configuration with External Hybrid

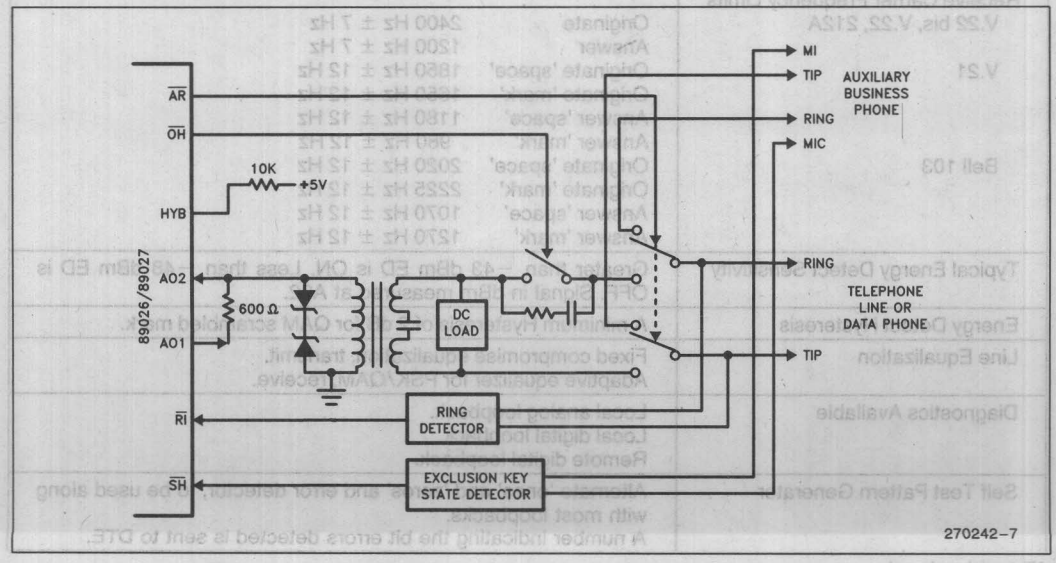


Figure 4. Typical Telephone Line Interface Using Internal Hybrid

Telephone Line Interface	Two wire full duplex over public switched network or 4 wire leased lines. On-chip hybrid and billing delay timers.	
Modulation	V.22 bis, 16 point QAM at 600 baud. V.22 and 212A, 4 point PSK at 600 baud. V.21 and 103, binary phase coherent FSK	
Output Spectral Shaping	Square root of 75% raised cosine, QAM/PSK.	
Transmit Carrier Frequencies		
V.22 bis, V.22, 212A	Originate	1200 Hz ± .01%
	Answer	2400 Hz ± .01%
V.21	Originate 'space'	1180 Hz ± .01%
	Originate 'mark'	980 Hz ± .01%
	Answer 'space'	1850 Hz ± .01%
	Answer 'mark'	1650 Hz ± .01%
Bell 103 mode	Originate 'space'	1070 Hz ± .01%
	Originate 'mark'	1270 Hz ± .01%
	Answer 'space'	2020 Hz ± .01%
	Answer 'mark'	2225 Hz ± .01%
Receive Carrier Frequency Limits		
V.22 bis, V.22, 212A	Originate	2400 Hz ± 7 Hz
	Answer	1200 Hz ± 7 Hz
V.21	Originate 'space'	1850 Hz ± 12 Hz
	Originate 'mark'	1650 Hz ± 12 Hz
	Answer 'space'	1180 Hz ± 12 Hz
	Answer 'mark'	980 Hz ± 12 Hz
Bell 103	Originate 'space'	2020 Hz ± 12 Hz
	Originate 'mark'	2225 Hz ± 12 Hz
	Answer 'space'	1070 Hz ± 12 Hz
	Answer 'mark'	1270 Hz ± 12 Hz
Typical Energy Detect Sensitivity	Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at A02.	
Energy Detect Hysteresis	A minimum Hysteresis of 2 dB for QAM scrambled mark.	
Line Equalization	Fixed compromise equalization, transmit. Adaptive equalizer for PSK/QAM, receive.	
Diagnostics Available	Local analog loopback. Local digital loopback. Remote digital loopback.	
Self Test Pattern Generator	Alternate 'ones' and 'zeros' and error detector, to be used along with most loopbacks. A number indicating the bit errors detected is sent to DTE.	

*External code only.

RECEIVER PERFORMANCE

Test Cases		Typical SNR for 10^{-5} BER Performance	
Data Mode	Rx Level (dBm)	Answer (dB)	Originate (dB)
V.22 bis Synchronous	−30	16	16.5
	−40	16.5	18
V.22/Bell 212A Synchronous	−30	6.5	6.5
	−40	6.5	6.5
V.21 Asynchronous	−30	9	7.5
	−40	9	8
Bell 103 Asynchronous	−30	10	11.5
	−40	10	11.5

Test Conditions:

—Recieve signal (Rx) measured at A02 (transmit level set at −9 dBm)

—Unconditioned 3002 line

—3 kHz Flat-band Noise

4

PERFORMANCE SPECIFICATIONS

Parameter	Min	Typ	Max	Units	Comments
DTMF Level		4.0		dBm	at AO1
DTMF Second Harmonic			−35	dB	HYB enabled into 600Ω
DTMF Twist (Balance)		3		dB	
DTMF Duration		100		ms	Software Controlled
Pulse Dialing Rate		10		pps	
Pulse Dialing Make/Break		39/61 33/67		% %	US UK, Hong Kong
Pulse Interdigit Interval		785		ms	
Billing Delay Interval			2.1	sec	
Guard Tone Frequency Amplitude Frequency Amplitude		540 −3 1800 −6		Hz dB Hz dB	referenced to High Channel transmit. QAM/PSK Modes Only
Dial Tone Detect Duration		3.0		sec	
Ringback Tone Detect Duration Cadence		 0.75 1.5		 sec	 Off/On Ratio
Busy Tone Detect Duration Cadence	 0.67	 0.2	 1.5	 sec	 Off/On Ratio

89026 OVERVIEW

The 89026 processor performs data manipulation, signal processing and user interface functions. It supports an external ROM, for user designed software. This option allows customer designed code to control the signal processing algorithms resident in the 89026. For example proprietary modem control and call progress management applications can be implemented using EPROMs or alternatively by having it burnt in the processor ROM (done so by Intel factory contracting). On-chip ROM is 8 Kbytes. A block diagram of 89026 is in Figure 5.

89026 contains a TTL compatible serial link to DTE/DCE equipment, along with a full complement of V.24/RS-232-C control signals. Alternatively, UART or USART may be used to directly transfer data to and from a microcomputer bus. The industry standard AT command set is supported by the 89026, facilitating communications compatibility between 89024 and most PC software written for the AT command set.

In the transmit operation, the 89026 synthesizes DTMF tones and the 300 bps FSK modem signal prior to transmitting them to the 89027 as digitized amplitude samples. During 1200 and 2400 bps operation, PSK and QAM is used to send 2 or 4 bits of information respectively at 600 baud to 89027. Since the QAM coding technique is an inherently synchronous transmission mechanism, during asynchronous QAM transmission, the asynchronous data is synchronized by adding or deleting stop bits. Following the synchronization process, the 89026 transmits digitized phase and amplitude samples to 89027 over a high speed serial link.

In the receive operation, the information is received by 89026 from 89027 as two signals which are 90 degrees phase shifted from each other. These analog signals are then digitized by the 89026's onboard A/D converter, and using DSP software algorithms the signals are gain adjusted, adaptively equalized for telephone line delay and amplitude distortion, and demodulated. Following the demodulation process by the 89026, the data is unscrambled, and if necessary, returned to asynchronous format.

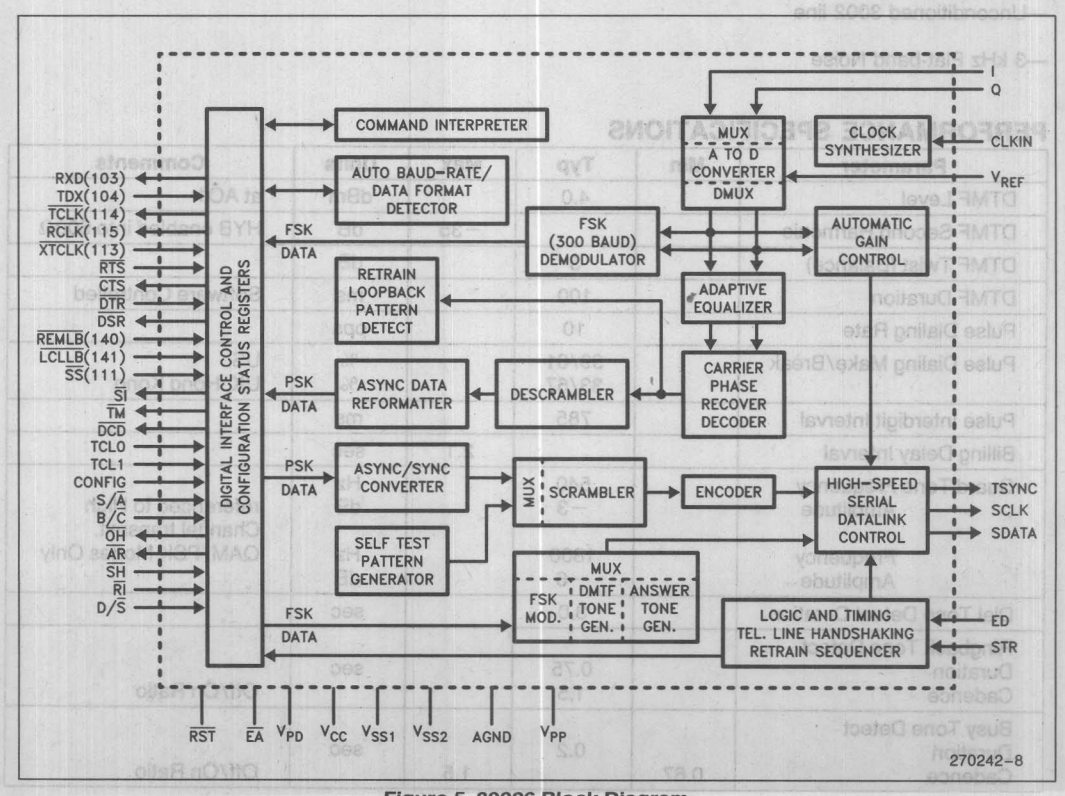


Figure 5. 89026 Block Diagram

89026 PINOUT

Symbol	Function (89026)	Direction	Pin No. 68 pin
CLKIN	12.96 MHz master clock from 89027	In	67
$\overline{RST}$	Chip reset (active low)	In	16
I	In-phase received signal	In	11
Q	Quadrature-phase received signal	In	10
STR	Symbol Timing from 89027	In	24
ED	Energy Detect input	In	9
TSYNC	Transmitter sync pulse to 89027	Out	35
SDATA	Serial Data to 89027	Out	17
SCLK	Serial Clock to 89027	Out	18
$\overline{OH}$	Off-Hook control to DAA	Out	33
$\overline{SH}$*	Switch-Hook from dataphone	In	44
$\overline{RI}$	Ring Indicator from DAA	In	42
$\overline{AR}$	Aux Relay control to DAA	Out	38
TCL1	NVRAM Data I/O	I/O	20
TCL0	NVRAM CLK	Out	19
B/$\overline{C}$*	103/V.21 default option	In	15
S/$\overline{A}$	NVRAM CE	Out	21
D/$\overline{S}$	Dumb/$\overline{Smart}$ mode select	In	6
CONFIG	Reserved for future use (V_{CC})(3)	In	8
$\overline{TM}$	Test Mode Indicator	Out	39
TXD	Transmitted data from DTE	In	27
RXD	Received data to DTE	Out	29
$\overline{RTS}$	Request to send from DTE	In	22
$\overline{CTS}$	Clear to Send to DTE	Out	23
$\overline{DSR}$	Data Set Ready to DTE	Out	30
$\overline{DCD}$	Data Carrier Detect to DTE	Out	31
$\overline{DTR}$	Data Terminal Ready from DTE	In	25
$\overline{RCLK}$	Received clock to DTE	Out	34
$\overline{TCLK}$	Transmit clock to DTE	Out	28
$\overline{XTCLK}$*	External timing clock from DTE	In	26
$\overline{SI}$	Speed Indicator to DTE	Out	32
$\overline{SS}$	(Note 4)	In	5
$\overline{REMLB}$*	Remote Loopback Command from DTE	In	7
$\overline{LCLLB}$*	Local Loopback Command from DTE	In	4
V_{CC}	Positive power supply (+5V)	+5V	1
V_{PD}	Ram back-up power (V_{CC})(3)	+5V	14
V_{REF}	A/D converter reference	+5V	13
V_{SS1}	Digital ground	GND	36
V_{SS2}	Digital ground	GND	68
AGND	Analog ground	AGND	12
V_{PP}	(NC)(2)	In	37
$\overline{EA}$	External Memory enable	In	2
AD0-AD15	External memory access address/data(5)	I/O	60-45
$\overline{AA}$	Auto Answer(5)	Out	60
$\overline{JS}$	*Jack Select*(5)	Out	59

*Available in external code only.

WR	External memory write	Out	40

NOTES:
1. Pins marked with (V_{SS}) must be connected to V_{SS}.
2. Pins marked with (NC) are to be left unconnected.
3. Pins marked with (V_{CC}) must be connected to V_{CC}.
4. $\overline{SS}$ pin reserved for future use.
5. With internal ROM enabled, AD0-AD1 are used as $\overline{AA}$ and $\overline{JS}$.
6. Pins with direction "In" must not be left floating.

89026 PIN DESCRIPTION

$\overline{XTCLK}$*

Transmitter timing from DTE, when external clock option is selected.

TXD

The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89026 samples this data on the rising edges of $\overline{TCLK}$.

$\overline{TCLK}$*

Clock output from 89026 as timing source for data exchange from DTE to modem. Serial data is read on the rising edges of the $\overline{TCLK}$. This output is High in asynchronous mode.

RXD

The serial data to DTE. 'Mark' is a logic High. In synchronous mode, the rising edge of $\overline{RCLK}$ occurs in the middle of RXD.

$\overline{RCLK}$*

Synchronous clock output. Rising edge of $\overline{RCLK}$ occurs in the middle of each RXD bit. This pin remains High in asynchronous mode.

V_{PP}

This function is not used and should not be connected.

*External code only.

$\overline{TM}$

A Low indicates maintenance condition in the modem.

$\overline{DCD}$

In async operation, $\overline{DCD}$ remains Low regardless of data carrier (default), or it can be programmed to indicate received carrier signal is within the required timing and amplitude limits. In sync operation Low indicates the received carrier signal is within the required timing and amplitude limits.

$\overline{DSR}$

Low indicates modem is off-hook, and it is in data transmission mode, and the answer tone is being exchanged. $\overline{CTS}$ Low indicates modem is prepared to accept data.

$\overline{RTS}$

In async mode $\overline{RTS}$ is ignored. Under command control, in sync mode $\overline{RTS}$ can be ignored, or the modem can respond with a Low on $\overline{CTS}$.

$\overline{DTR}$

&D0 command will cause the modem to ignore $\overline{DTR}$. For &D1 the modem assumes the asynchronous command state on a Low to High transition of the $\overline{DTR}$ circuit. The &D2 command does the same as &D1 except the state of $\overline{DTR}$ will enable/disable auto answer. A Low to High transition of $\overline{DTR}$ after the &D3 command will cause the modem to assume the initialization state.

B/$\overline{C}$*

Low configures the modem to CCITT V.21. High will configure the modem to Bell 103, when at 300 bps speed. This pin only affects the modem in FSK operation.

TCL1, TCL0

These pins are used as the serial clock and data for interface to an NVRAM. Refer to Figure 3. TCL0 is used to output a clock and serial data is transferred on TCL1.

$\overline{AR}$

This Auxiliary relay control is for switching a relay for voice or data calls. High is voice, Low is data.

$\overline{RI}$

A Low signal from DAA indicates line ringing. This input is ignored when the modem is configured for leased line. This signal should follow the ring cadence.

$\overline{OH}$

Low sets an off hook condition, high sets an on hook. When dialing, this signal is used to pulse dial the line.

$\overline{SH}$*

Used as a telephone voice to data switch or vice versa. Any logic level transition will toggle the modem state. This input is ignored, if a software command attempts to switch the modem between voice and data.

$\overline{AA}$

Used as an indicator for Auto Answer status and Ring indicator. Active low.

$\overline{LCLLB}$*

A Low will set the modem in the local analog loopback test mode. Logic Low levels applied simultaneously to $\overline{REMLB}$ and $\overline{LCLLB}$ pins, sets the modem to the local digital loopback.

$\overline{REMLB}$*

A logic Low on this pin initiates a remote loopback condition.

$\overline{SI}$

Selects one of the two data rates or ranges of rates in the DTE to correspond to the rate in modem. Low selects the higher rate (2400 CCITT/1200 BELL) or range of rates. High selects the Low rate or range of rates.

D/$\overline{S}$

A Low on this pin will indicate the smart mode which will respond to all commands. A High will ignore all commands.

V_{REF}

Voltage reference for the analog to digital converter should be connected to the 89027 AVcc.

V_{PD}

The internal RAM power down supply voltage to be connected to 5 Volts during normal operation.

S/$\overline{A}$

The function of this pin is re-defined as external NVRAM CE.

CONFIG

Reserved for future use. This signal should be pulled high.

$\overline{EA}$

When High, memory access from address 2000H to 4000H are directed to on-chip ROM. When Low, all memory access is directed to off-chip memory.

$\overline{JS}$

Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

*External code only

4

89026 ABSOLUTE MAXIMUM RATINGS**

Temperature Under Bias . . . . . 0°C to +70°C

Storage Temperature . . . . . −40°C to +125°C

Voltage from Any Pin to V_{SS} or AGND . . . . . −0.3V to +7.0V

Average Output Current from Any Pin . . . . . 10 mA

Power Dissipation . . . . . 1.5 Watts

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
TA	Ambient Temperature Under Bias	0	+70	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V
V_{REF}	Analog Supply Voltage	4.75	5.25	V
FREQ	CLKIN Frequency 12.96 MHz	−0.01%	+0.01%	
V_{PD}	Power-Down Supply Voltage	4.75	5.25	V

NOTE:
The AGND and V_{SS} on both the 89026 and the 89027 must be nominally at the same potential.

D.C. CHARACTERISTICS

Test Conditions: V_{CC}, V_{REF}, V_{PD}, V_{PP}, V_{EA} = 5.0V ±0.25V; F_{OSC} = 12.96 MHz; T_A = 0°C to 70°C, V_{SS}, AGND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
I_{CC}	V_{CC} Supply Current (0°C ≤ T_A ≤ 70°C)		240	mA	All Outputs Disconnected
I_{CC1}	V_{CC} Supply Current (T_A = 70°C)		185	mA	
I_{PD}	V_{PD} Supply Current		1	mA	Normal Operation and Power-Down
I_{REF}	V_{REF} Supply Current		8	mA	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage (Except $\overline{RESET}$, NMI, CLKIN)	2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage, $\overline{RESET}$ Rising	2.4	V_{CC} + 0.5	V	
V_{IH2}	Input High Voltage, $\overline{RESET}$ Falling Hysteresis	2.1	V_{CC} + 0.5	V	
V_{IH3}	Input High Voltage, NMI, CLKIN	2.2	V_{CC} + 0.5	V	
I_{LI}	Input Leakage Current(1)		±10	μA	V_{IN} = 0 to V_{CC}
I_{LI1}	D.C. Input Leakage Current(2)		+3	μA	V_{IN} = 0 to V_{CC}
I_{IH}	Input High Current to $\overline{EA}$		100	μA	V_{IH} = 2.4V
I_{IL}	Input Low Current(3)		−125	μA	V_{IL} = 0.45V
I_{IL1}	Input Low Current to $\overline{RESET}$	−0.25	−2	mA	V_{IL} = 0.45V
I_{IL2}	Input Low Current, READY, BUSWIDTH(4)		−50	μA	V_{IL} = 0.45V
V_{OL}	Output Low Voltage(5)		0.45	V	I_{OL} = 0.8 mA
V_{OL1}	Output Low Voltage(5)		0.75	V	I_{OL} = 2.0 mA
V_{OL2}	Output Low Voltage(6) $\overline{RESET}$ and Bus/Control Pins		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage(5)	2.4		V	I_{OH} = −20 μA
V_{OH1}	Output High Voltage on Bus Control Pins(7)	2.4		V	I_{OH} = −200 μA
I_{OH3}	Output High Current on $\overline{RESET}$	−50		μA	V_{OH} = 2.4V
C_S	Pin Capacitance (Any Pin to V_{SS})		10	pF	f_{TEST} = 1.0 MHz

NOTES:
1. STR, $\overline{DTR}$, $\overline{XTCLK}$, TXD
2. S/$\overline{D}$, $\overline{SS}$, $\overline{REMLB}$, $\overline{LCLLB}$, I, Q, CONFIG, ED
3. $\overline{TCLK1}$, $\overline{RTS}$
4. Also, B/$\overline{C}$, $\overline{SH}$, $\overline{RI}$
5. TCL0, S/$\overline{A}$, $\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, SI, $\overline{OH}$, $\overline{AR}$
6. SCLK, SDATA, $\overline{TM}$, $\overline{TCLK}$, RXD, $\overline{RCLK}$, TSYNC
7. Bus/Control pins include CLKOUT, ALE, $\overline{BHE}$, $\overline{RD}$, $\overline{WR}$, INST and AD0–15.

AC CHARACTERISTICS (V_{CC}, V_{PD} = 4.75 to 5.25 Volts; T_A = 0°C to 70°C; CLKIN = 12.96 MHz)

Test Conditions: Load capacitance on output pins = 80 pF
T_{OSC} = 1/12.96 MHz

TIMING REQUIREMENTS (Other system components must meet these specs.)

Symbol	Parameter	Min	Max	Units
T_{CLYX}	READY Hold after CLKOUT Edge	0		ns
T_{LLYV}	End of ALE/$\overline{ADV}$ to READY Valid		2Tosc − 70	ns
T_{LLYH}	End of ALE/$\overline{ADV}$ to READY High	2Tosc + 40	4Tosc − 80	ns
T_{YLYH}	Non-Ready Time		1000	ns
T_{AVDV}(1)	Address Valid to Input Data Valid		5Tosc − 120	ns
T_{RLDV}	$\overline{RD}$ Active to Input Data Valid		3Tosc − 100	ns
T_{RHDX}	Data Hold after $\overline{RD}$ Inactive	0		ns
T_{RHDZ}	$\overline{RD}$ Inactive to Input Data Float	0	Tosc − 25	ns
T_{AVGV}(1)	Address Valid to BUSWIDTH Valid		2Tosc − 125	ns
T_{LLGX}	BUSWIDTH Hold after ALE/$\overline{ADV}$ Low	Tosc + 40		ns
T_{LLGV}	ALE/$\overline{ADV}$ Low to BUSWIDTH Valid		Tosc − 75	ns

NOTE:
1. The term "Address Valid" applies to AD0–15, $\overline{BHE}$ and INST.

TIMING RESPONSES

Symbol	Parameter	Min	Max	Units
F_{CLKIN}	Oscillator Frequency	12.95870	12.96129	MHz
T_{OSC}	Oscillator Period	1/$F_{CLKIN(MAX)}$	1/$F_{CLKIN(MIN)}$	ns
T_{OHCH}	Rising Edge to Clock Rising Edge	0	120	ns
T_{CHCH}	CLKOUT Period(2)	3Tosc(2)	3Tosc(2)	ns
T_{CHCL}	CLKOUT High Time	Tosc − 35	Tosc + 10	ns
T_{CLLH}	CLKOUT Low to ALE High	−20	+25	ns
T_{LLCH}	ALE/$\overline{ADV}$ Low to CLKOUT High	Tosc − 25	Tosc + 45	ns
T_{LHLL}	ALE/$\overline{ADV}$ High Time	Tosc − 30	Tosc + 35(3)	ns
T_{AVLL}(4)	Address Setup to End of ALE/$\overline{ADV}$	Tosc − 50		ns
T_{RLAZ}	$\overline{RD}$ or $\overline{WR}$ Low to Address Float	Typ. = 0	10	ns
T_{LLRL}	End of ALE/$\overline{ADV}$ to $\overline{RD}$ or $\overline{WR}$ Active	Tosc − 40		ns
T_{LLAX}(5)	Address Hold after End of ALE/$\overline{ADV}$	Tosc − 40		ns
T_{WLWH}	$\overline{WR}$ Pulse Width	3Tosc − 35		ns
T_{QVWH}	Output Data Valid to End of $\overline{WR}$/$\overline{WRL}$/$\overline{WRH}$	3Tosc − 60		ns
T_{WHQX}	Output Data Hold after $\overline{WR}$/$\overline{WRL}$/$\overline{WRH}$	Tosc − 50		ns
T_{WHLH}	End of $\overline{WR}$/$\overline{WRL}$/$\overline{WRH}$ to ALE/$\overline{ADV}$ High	Tosc − 75		ns
T_{RLRH}	$\overline{RD}$ Pulse Width	3Tosc − 30		ns
T_{RHLH}	End of $\overline{RD}$ to ALE/$\overline{ADV}$ High	Tosc − 45		ns

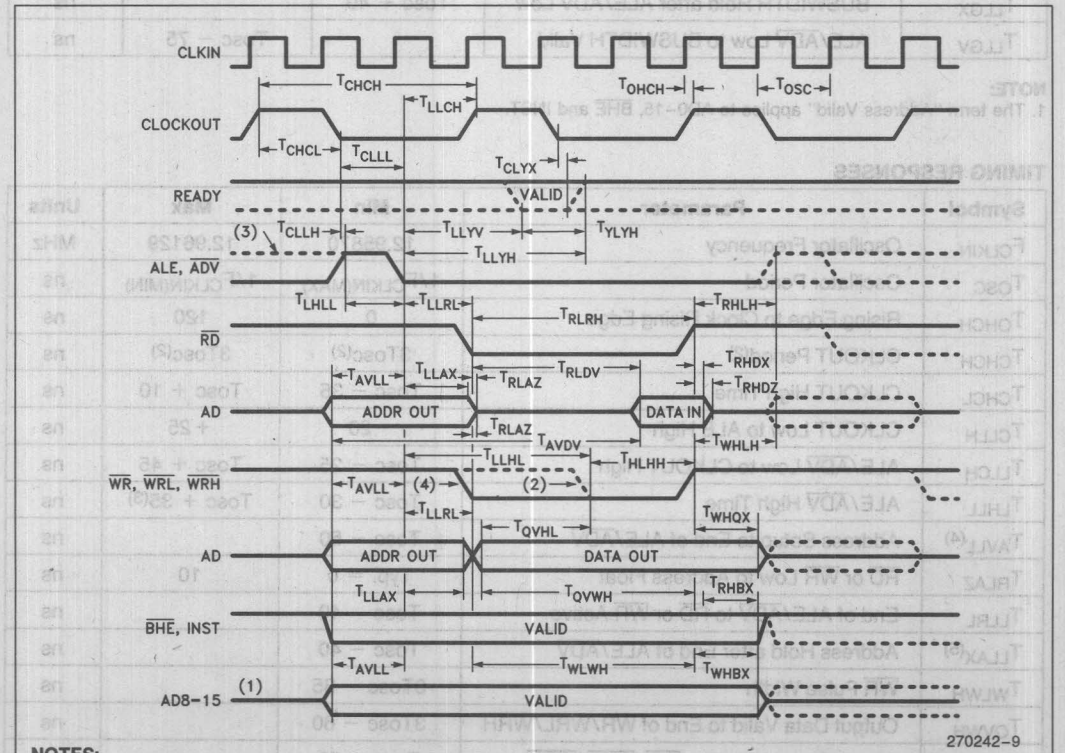
4

TIMING RESPONSES (Continued)

Symbol	Parameter	Min	Max	Units
T_{CLLL}	CLKOUT to Low ALE/$\overline{ADV}$ Low	Tosc − 40	Tosc + 35	ns
T_{RHBX}	$\overline{RD}$ High to INST, $\overline{BHE}$, AD8–15 Inactive	Tosc − 25	Tosc + 30	ns
T_{WHBX}	$\overline{WR}$ High to INST, $\overline{BHE}$, AD8–15 Inactive	Tosc − 50	Tosc + 100	ns
T_{HLHH}	$\overline{WRL}$, $\overline{WRH}$ Low to $\overline{WRL}$, $\overline{WRH}$ High	2Tosc − 35	2Tosc + 40	ns
T_{LLHL}	ALE/$\overline{ADV}$ Low to $\overline{WRL}$, $\overline{WRH}$ Low	2Tosc − 30	2Tosc + 55	ns
T_{QVHL}	Output Data Valid to $\overline{WRL}$, $\overline{WRH}$ Low	Tosc − 60		ns

NOTES:
1. If more than one wait state is desired, add 3Tosc for each additional wait state.
2. CLKOUT is directly generated as a divide by 3 of the oscillator. The period will be 3Tosc ±10 ns if Tosc is constant and the rise and fall times are less than 10 ns.
3. Max spec applies only to ALE. Min spec applies to both ALE and $\overline{ADV}$.
4. The term "Address Valid" applies to AD0–15, $\overline{BHE}$ and INST.
5. The term "Address" in this definition applies to AD0–7 for 8-bit cycles, and AD0–15 for 16-bit cycles.

WAVEFORM



NOTES:
1. 8-bit bus only
2. 8-bit or 16-bit bus and write strobe mode selected.
3. When $\overline{ADV}$ selected.
4. 8- or 16-bit bus and no write strobe mode selected.

Figure 6. Bus Signal Timings

89027 OVERVIEW

The 89027 is a 28 pin CHMOS analog front end device, which performs most of the complex filtering functions required in modem transmitters and receivers. A general block diagram of this chip is provided in Figure 7. Most of the analog signal processing functions in this chip are implemented with CHMOS switched capacitor technology. The 89027 functions are controlled by 89026, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89026. The 89027 converts the signal to its analog equivalent, filters it, and transmits it to the telephone line. For QAM transmission, the signal constellation points are transferred to the 89027. This information is modulated into an analog signal, passed through spectral shaping filters, combined with the necessary guard tone, smoothed by a low pass filter, and transmitted to the line. Prior to transmitting either FSK or QAM signals to the telephone line, the 89027 adjusts the signal gain through an on-board programmable gain amplifier.

During the receive operation, the received FSK and QAM signals are passed through anti-alias filters, bandsplit filters, automatic gain control and carrier detect circuits, a Hilbert transform filter, and the output sent to the 89026 processor as analog signals.

Other functions provided by the 89027 are: an on-board two wire to four wire circuit with disable capability, an audio monitor output with software configurable gain, and a programmable gain transmit signal.

The 89027 is available in 28 pin plastic DIP and PLCC packages.

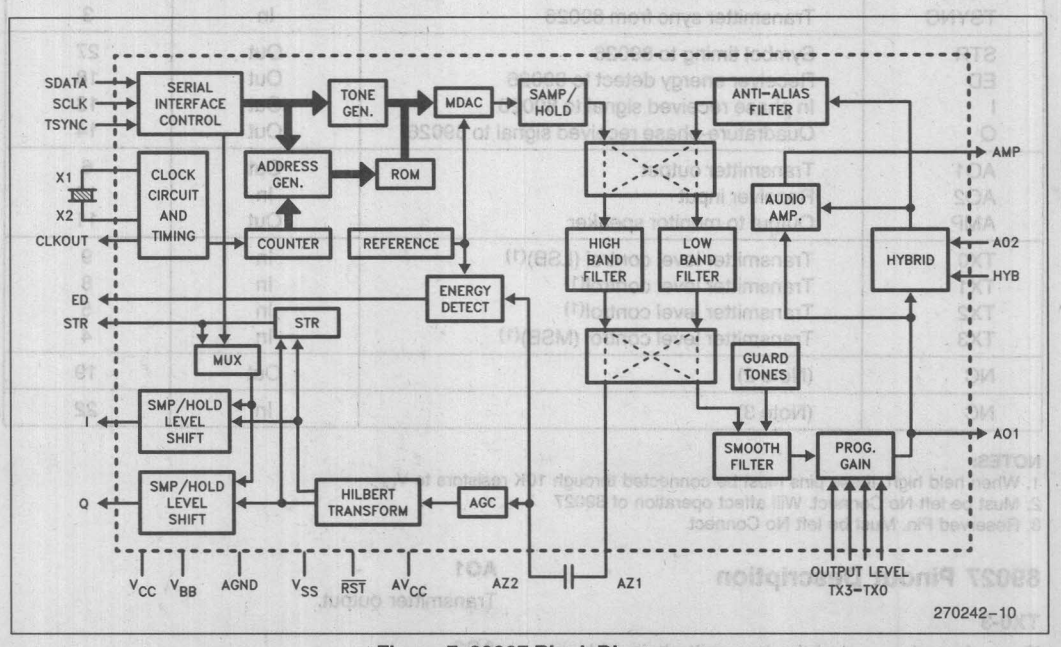


Figure 7. 89027 Block Diagram

89027 PINOUT

Symbol	Function (89027)	Direction	Pin No.
V_{CC}	Positive Power Supply (Digital)	+5V	28
V_{BB}	Negative Power Supply	−5V	15
V_{SS}	Digital Ground	DGND	24
AGND	Analog Ground	AGND	21
AV_{CC}	Positive Power Supply (Analog)	+5	7
X1	Xtal Oscillator	In	23
X2	Xtal Oscillator	Out	25
CLKOUT	12.96 MHz Clock Output to 89026	Out	26
$\overline{RST}$	Chip reset (active low)(1)	In	20
HYB	Enable on-chip hybrid(1)	In	10
AZ1	Auto-zero capacitor	Out	16
AZ2	Auto-zero capacitor	In	17
SDATA	Serial data from 89026	In	2
SCLK	Serial clock from 89026	In	1
TSYNC	Transmitter sync from 89026	In	3
STR	Symbol timing to 89026	Out	27
ED	Receiver energy detect to 89026	Out	18
I	In phase received signal to 89026	Out	13
Q	Quadrature-phase received signal to 89026	Out	14
AO1	Transmitter output	Out	6
AO2	Receiver input	In	12
AMP	Output to monitor speaker	Out	11
TX0	Transmitter level control (LSB)(1)	In	9
TX1	Transmitter level control(1)	In	8
TX2	Transmitter level control(1)	In	5
TX3	Transmitter level control (MSB)(1)	In	4
NC	(Note 2)	Out	19
NC	(Note 3)	In	22

NOTES:
1. When held high, these pins must be connected through 10K resistors to V_{CC}.
2. Must be left No Connect. Will affect operation of 89027
3. Reserved Pin. Must be left No Connect.

89027 Pinout Description

TX0-3

These four pins control the transmitted signal level. Refer to Transmit Level Table.

HYB

This pin enables the on-chip hybrid. A line impedance matching network must be connected between AO1 and AO2 when HYB is enabled. If HYB is disabled and an external 4W/2W hybrid is used, the hybrid receive path must be amplified by 6 dB.

AO1

Transmitter output.

AO2

Receiver input.

AMP

This output can be used to monitor the call progress tones and operation of the line.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −0 to +70°C
Storage Temperature ..... −40 to +125°C
All Input and Output Voltages with Respect to V_{BB} ..... −0.3V to +13.0V
All Input and Output Voltages with Respect to V_{CC} & AV_{CC} ..... −13.0V to 0.3V
Power Dissipation 1.35W
Voltage with Respect to V_{SS}(1) ..... −0.3V to 6.5V

NOTE:
1. Applies to pins SCLK, SDATA, TSYNC, $\overline{RST}$, HYB, TX0–TX3 only.

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

POWER DISSIPATION

Ambient Temp = 0°C to 70°C, V_{CC} = AV_{CC} = 5V ±5%, V_{SS} = AGND = 0V.

Symbol	Parameter	Min	Typ	Max	Units
$AIcc_1$	AV_{CC} Operating Current		15	21	mA
Icc_1	V_{CC} Operating Current		5	6	mA
Ibb_1	V_{BB} Operating Current		−15	−21	mA
AIccs	AV_{CC} Standby Current		0.2	1	mA
Iccs	V_{CC} Standby Current		5	6	mA
Ibbs	V_{BB} Standby Current		−0.6	−2	mA
Pdo	Operating Power Dissipation		175	250	mW
Pds	Standby Power Dissipation		30	50	mW

DC CHARACTERISTICS

(Ta = 0°C to 70°C, AV_{CC} = V_{CC} = 5V ±5%, V_{BB} = 5V ±5%, AGND = V_{SS} = 0V), supply voltage must be at the same potential as the 89026 power supply. Typical Values are for Ta = 25°C and nominal power supply values. V_{CC}, AV_{CC} and 89026 V_{REF} must be nominally at the same potential.

Inputs: TX0, TX1, TX2, TX3, HYB, $\overline{RST}$
Outputs: CLKOUT

Symbol	Parameter	Min	Max	Units	Test Condition
Iil	Input Leakage Current	−10	+10	µA	$V_{SS} \le Vin \le V_{CC}$
Vil	Input Low Voltage	V_{SS}	0.8	V	
Vih	Input High Voltage	2.0	V_{CC}	V	
Vol	Output Low Voltage		0.4	V	Iol ≥ −1.6mA,1 TTL load
Voh	Output High Voltage	2.4		V	Ioh ≤ 50µa, 1 TTL load
V_{col}	CLKOUT Low Voltage		0.4	V	Load Capacitance = 60 pF
Vcoh	CLKOUT High Voltage	0.7 V_{CC}		V	Load Capacitance = 60 pF

4

AC CHARACTERISTICS ($T_a = 25°C$, $V_{CC} = AV_{CC} = 5V$, $V_{SS} = AGND = 0V$, $V_{BB} = -5V$)

ANALOG INPUTS: AO2

Parameter	Min	Typ	Max	Units	Test Condition
AO2 Receive Signal			−9	dBm	Hybrid Enabled
AO2 Input Resistance		10		MOhms	$-2.5V < Vin < +2.5V$
AO2 Allowed DC offset	−30		+30	mV	Relative to AGND

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF
Tolerance = ±20 %
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.

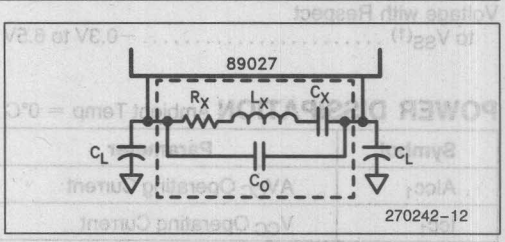


Figure 8. Crystal Equivalent Circuit

CRYSTAL REQUIREMENTS

Parameter	Min	Typ	Max	Unit	Comments
Frequency Accuracy (0°C–70°C)	−0.0035%	12.96	+0.0035%	MHz	Refer to Figure 8
Rx		10	16	Ohms	
Cx		0.024		pF	
Co	5.1	5.6	6.1	pF	
C_L	−5%	33	+5%	pF	2 Load Capacitors

Crystal Type: Parallel Resonant

ANALOG OUTPUTS: A01, AMP

Parameter	Min	Typ	Max	Units	Comments
Load Resistance					
AO1	600			Ohms	
AMP	10			KOhms	
Load Capacitance					
AMP			100	pF	
Audio Amp Gain		−9		dB	Max
AO1 to Amp		−18		dB	Mid
		−26		dB	Min Software
		−70		dB	Off Selectable
Audio Amp Gain (1)		+12		dB	Max
AO2 to Amp		+3		dB	Mid
		−4		dB	Min Software
		−60		dB	Off Selectable

NOTE:
1. Assumes on-chip hybrid is enabled. When on-chip hybrid is disabled, gain with respect to AO2 is reduced by 6 dB.

MODEM REFERENCE MANUAL

The Modem Reference Manual details design information for the 89024 Modem Chip Set. It provides descriptions and specifications of the two chips comprising the 89024, the 89026 and the 89027. In addition, it describes the control interface between the two chips.

The reference manual also provides a full description of all the AT commands and S-registers supported by the Modem Chip Set.

4

89024 REVISION -005 HISTORY

The following differences exist between Rev. -004 and this version of the data sheet:

1. The internal code has been revised from 1.6 to 2.0. The following are the major changes and improvements:
   - i) Expanded NVRAM (9346) support which allows storage and retrieval of four phone numbers and two user profiles. Commands supported are &n, DS, &Wn and Zn.
   - ii) S11 DTMF duration and spacing register support
   - iii) The B command selects between V.21 and Bell 103 modes for 300 bps connections. The B/C pin on 89026 is ignored.
   - iv) Factory defaults have been changed from &CO &DO to &C1, &D2 respectively.
   - v) Expanded functionality of ATO command. ATO from the call program mode causes the modem to go into originate mode instead of answer mode. When off line, ATO gives "NO CARRIER" response, instead of going on line.
   - vi) Changed response to test modes.
   - vii) DTR transition received during idle state will not generate any result codes, and will not have any effect on the modem.
   - viii) 600 bps operation is not supported.
   - ix) Switch Hook (SH) option is not supported.
   - x) LCLLB and REMLB loopback pins are not supported.
2. Receiver BER performance table has been expanded.
3. Frequency and Phase jitter specs have been removed.
4. 89024 Reference Manual has been superseded by the modem Reference Manual. This is a single manual for Intel's modem product line.

89C024LT
ERROR CORRECTING LAPTOP MODEM CHIP SET

- CHMOS for Low Operating Power
- Low Standby Power Requirements
- Minimum Chip Count for Small size
- MNP* Operation through Class 4 for Error Correction
- MNP Class 5 Data Compression and DTE Interface Rates of 9600 bps Can Provide Increased Throughput up to 4:1
- Serial Command Set Compatible with Hayes** Smartmodem 2400**
- V.22 bis, V.22 A/B, V.21, Bell 212A, and Bell 103 Compatible
- For Public Switched Telephone Network and Unconditioned Leased Line Applications
- Automatically Detects Remote Modem Type and Data Rate
- On-Chip Hybrid
- DTMF and Pulse Dialing
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Simple Serial Interface to External NVRAM
- Automatic Speed Matching in MNP and Normal Modes
- Hardware and Software Flow Control
- Analog/Digital Loopback Diagnostics with Mark/Space Pattern Generation and Error Detection
- Telephone Line Audio Monitor Output
- Full Set of Control Signals for DAA Interface
- International Call Progress Tone Detection Capabilities
- Automatic Adaptive Equalization
- Synchronous Modes
- Easily Customized Command Set and Features
- Intel's MNP Software Co-Developed with R. Scott Associates***

4

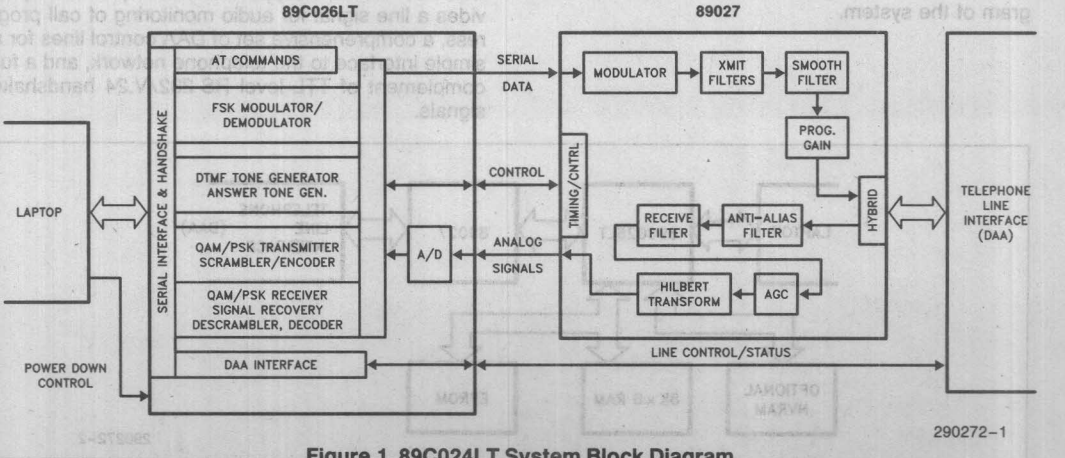


Figure 1. 89C024LT System Block Diagram

*MNP is a registered trademark of Microcom, Inc.
**Hayes, Smartmodem 2400, and Smartcom II are registered trademarks of Hayes Microcomputer Products, Inc.
***R. Scott Associates, Inc., 5711 Six Forks Road, Suite 301, Raleigh, North Carolina 27609, (919) 846-7171

October 1990
Order Number: 290272-002

GENERAL DESCRIPTION

Intel 89C024LT is a highly integrated, low power, error correcting laptop modem chip set. This two chip solution is composed of the 89027 Analog Front End and 89C026LT microcontroller. At 12.96 MHz the microcontroller is capable of executing error correction and data compression routines. The system is compatible with the following CCITT and BELL standards.

- CCITT V.22 bis
  2400 bps sync and async
  1200 bps sync and async
- CCITT V.22 A & B
  1200 bps sync and async
- CCITT V.21
  0 to 300 bps anisochronous
- BELL 212A
  1200 bps sync and async
  300 bps fall-back mode
- BELL 103
  0 to 300 bps anisochronous

The 89C024LT system consists of a 16 bit application specific processor (89C026LT) and an analog front end device (89027). The 89C026LT processor performs all "Digital Signal Processing" algorithm execution for processing the modem signals, as well as providing all modem control functions typically performed by an external processor. The analog front end provides for 2 wire and 4 wire telephone line interface, D/A conversion, and most of the complex filtering functions required in QAM/PSK/FSK modems. Refer to Figure 1 for a simplified block diagram of the system.

In laptop modem applications, the 89C024LT chip set along with a Data Access Arrangement (DAA), a single 8-bit EPROM, and an 8K x 8 static RAM represent the circuitry required for implementing an auto-dial, auto-answer, 300 to 2400 bps, MNP class 5 full duplex Hayes compatible intelligent modem. Refer to Figure 2 for a block diagram of this application.

A complete set of Industry Standard AT commands is provided for modem configuration and user interface. Additional commands have been implemented for power down modes and MNP feature control. Virtually all PC software written for the Hayes Smartmodem 2400 can also be used with this chip set. Alternatively, in applications where user proprietary modem control commands and features are desired, the user can replace the 89C024LT command module with custom proprietary software.

The 89C024LT has a set of default features. Upon power up, the modem configuration will be in accordance with these default options, unless a different configuration has been saved in optional external NVRAM with the &W command.

The 89C024LT modem has built in auto-dialing and auto-answering capabilities. It can be configured to the proper line signaling mode (Tone or Pulse), and to the type (CCITT or Bell) and speed of the calling or answering modem. It can also detect and identify call set-up signals of telephone networks, allowing unattended data call operation.

A full set of diagnostic loop-test features compatible with CCITT V.54 is supported. The chip set also provides a line signal for audio monitoring of call progress, a comprehensive set of DAA control lines for a simple interface to the telephone network, and a full complement of TTL level RS-232/V.24 handshake signals.

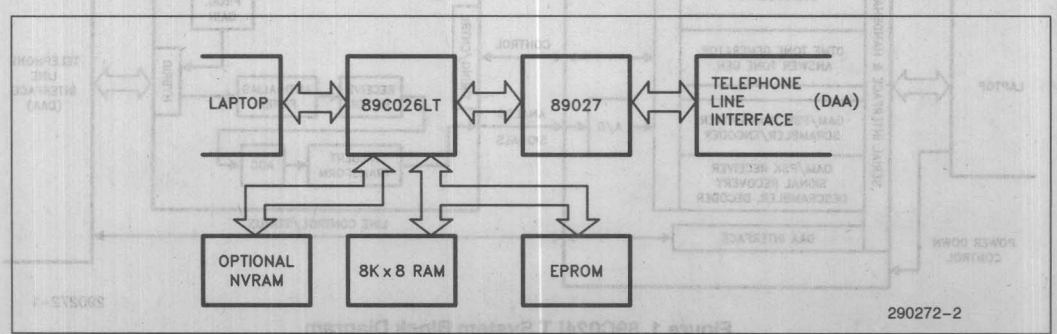


Figure 2. 89C024LT Laptop Modem Application

PACKAGING

89027 is available in PLCC and standard plastic DIP packages. The 89C026LT is available in a PLCC package. Packages are shown from top view, looking down on component side of PC board.

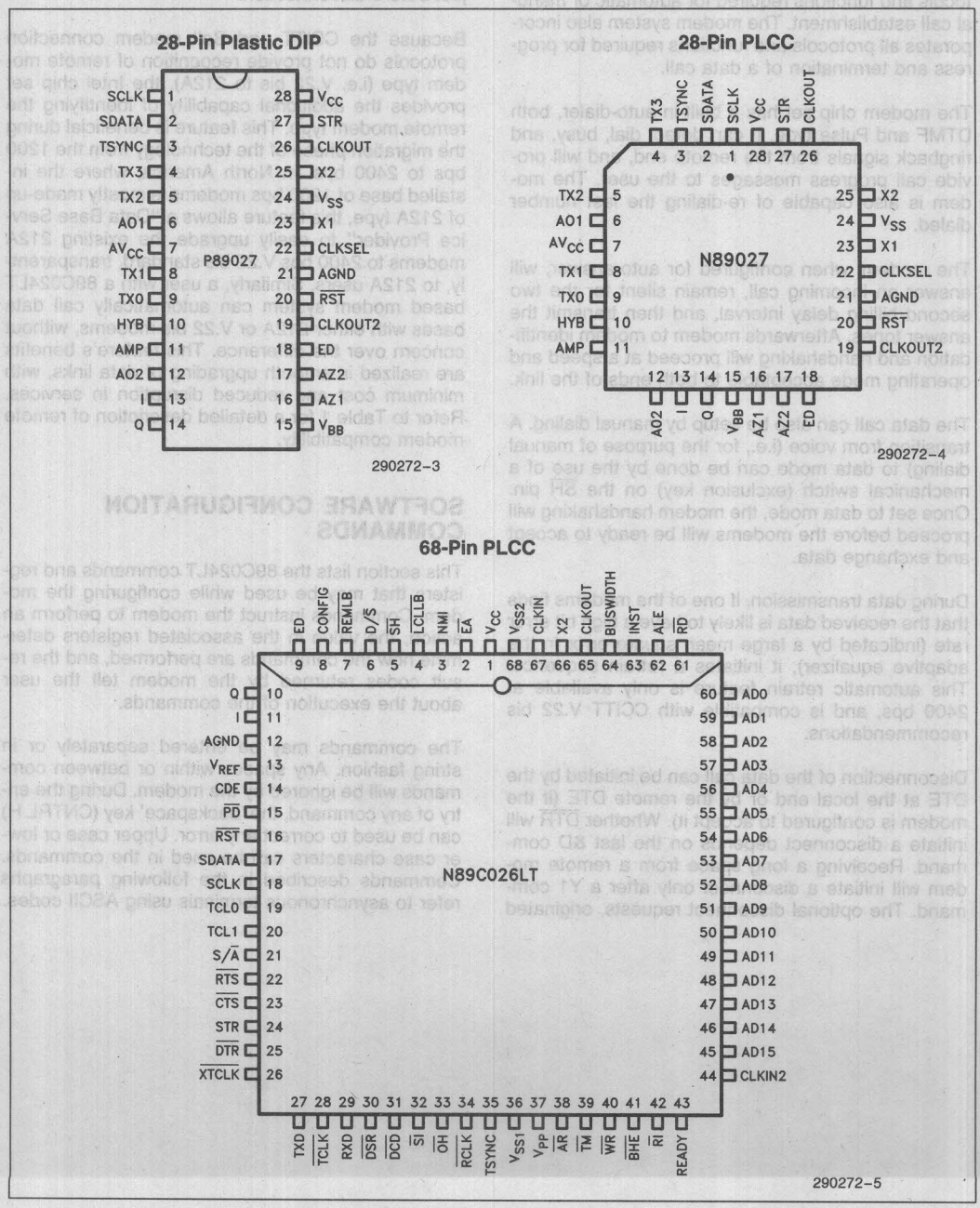


Figure 3. Device Packages

CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89C024LT modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip set has a built-in auto-dialer, both DTMF and Pulse type. It can detect dial, busy, and ringback signals from the remote end, and will provide call progress messages to the user. The modem is also capable of re-dialing the last number dialed.

The modem, when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, and then transmit the answer tones. Afterwards modem to modem identification and handshaking will proceed at a speed and operating mode acceptable to both ends of the link.

The data call can also be setup by manual dialing. A transition from voice (i.e., for the purpose of manual dialing) to data mode can be done by the use of a mechanical switch (exclusion key) on the $\overline{SH}$ pin. Once set to data mode, the modem handshaking will proceed before the modems will be ready to accept and exchange data.

During data transmission, if one of the modems finds that the received data is likely to have a high bit error rate (indicated by a large mean square error in the adaptive equalizer), it initiates a retrain sequence. This automatic retrain feature is only available at 2400 bps, and is compatible with CCITT V.22 bis recommendations.

Disconnection of the data call can be initiated by the DTE at the local end or by the remote DTE (if the modem is configured to accept it). Whether $\overline{DTR}$ will initiate a disconnect depends on the last &D command. Receiving a long space from a remote modem will initiate a disconnect only after a Y1 command. The optional disconnect requests, originated by the remote modem, are of two types, (1) disconnect when receiving long-space, and (2) disconnect when received carrier is dropped. The modem chip set can also be configured to transmit "long-space" just before disconnection.

Because the CCITT and Bell modem connection protocols do not provide recognition of remote modem type (i.e. V.22 bis to 212A), the Intel chip set provides the additional capability of identifying the remote modem type. This feature is beneficial during the migration phase of the technology from the 1200 bps to 2400 bps. In North America, where the installed base of 1200 bps modems is mostly made-up of 212A type, this feature allows a "Data Base Service Provider" to easily upgrade the existing 212A modems to 2400 bps V.22 bis standard, transparently, to 212A users. Similarly, a user with a 89C024LT based modem system can automatically call data bases with either 212A or V.22 bis modems, without concern over the difference. This feature's benefits are realized in smooth upgrading of data links, with minimum cost and reduced disruption in services. Refer to Table 1 for a detailed description of remote modem compatibility.

SOFTWARE CONFIGURATION COMMANDS

This section lists the 89C024LT commands and registers that may be used while configuring the modem. Commands instruct the modem to perform an action, the value in the associated registers determine how the commands are performed, and the result codes returned by the modem tell the user about the execution of the commands.

The commands may be entered separately or in string fashion. Any spaces within or between commands will be ignored by the modem. During the entry of any command, the 'backspace' key (CNTRL H) can be used to correct any error. Upper case or lower case characters can be used in the commands. Commands described in the following paragraphs refer to asynchronous terminals using ASCII codes.

Table 1. Remote Modem Compatibility

Originating 89C024LT Modem		Answering Modem Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	300	—	300*	300*
	1200	1200*	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	1200*	1200	—	1200	1200
	2400	1200*	1200		1200	2400

Answering 89C024LT Modem		Originating Modem Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	1200	—	1200	1200
	1200	300	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	300*	1200	—	1200	1200
	2400	300*	1200	—	1200	2400

* These connection data rates are obtained when connecting 89C024LT based modems end to end. The same results may not be obtained when a 89C024LT based modem is connected to other modems.

Command Set

AT	Attention code.
A	Go off-hook in answer mode
A/	Repeat previous command string
Bn	BELL/CCITT Protocol Compatibility at 300 and 1200 bps
Ds	The dialing commands (0-9 A B C D * # P R T S W , ; @)
En	Echo command (En)
Hn	Switch-Hook Control If &J1 option is selected, H1 will also switch the auxiliary relay
In	Request Product Code and Checksum
Ln	Speaker Volume
Mn	Monitor On/Off
O	On-Line
Qn	Result Codes
Sn=x	Write S Register
Sn?	Read S Register
Vn	Enable Short-Form Result Codes
Xn	Enable Extended Result Code
Yn	Enable Long Space Disconnect
Z	Fetch Configuration Profile
+++	The Default Escape Code

MNP Feature Control Command Set

\An	Maximum MNP Block Size
%An	Set Auto-Reliable Fallback Character
\Bn	Transmit Break
\Cn	Set Auto-Reliable Buffer
\Gn	Set Modem Port Flow Control
\Jn	Bits per Second Rate Adjust
\Kn	Set Break Control
\Nn	Set Operating Mode
\O	Originate Reliable Link
\Qn	Set Serial Port Flow Control
\Tn	Set Inactivity Timer
\U	Accept Reliable Link
\Vn	Modify Result Code Form
\Xn	Set XON/XOFF Pass-Through
\Y	Switch to Reliable Mode
\Z	Switch to Normal Mode

& Command Set

&Cn	DCD Options
&Dn	DTR Options
&Fn	Fetch Factory Configuration Profile
&Gn	Guard Tone
&Jn	Telephone Jack Selection
&Ln	Leased/Dial-up Line Selection
&Mn	Async/Sync Mode Selection
&Pn	Make/Break Pulse Ratio
&Rn	RTS/CTS Options
&Sn	DSR Options
&Tn	Test Commands
&Wn	Write Configuration to Non Volatile Memory
&Xn	Sync Clock Source
&Yn	Default NVRAM Profile Select
&Zn	Store Telephone Number

+ Command Set

+En	Disable/Enable Power Down
+Tn	Time to Power Down

4

CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to a special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

S0*	Ring to Answer
S1	Ring Count. (Read Only)
S2	Escape Code Character
S3	Carriage Return Character
S4	Line Feed Character
S5	Back Space Character
S6	Wait for Dial Tone
S7	Wait for Data Carrier
S8	Pause Time for the Comma Dial Modifier
S9	Carrier Detect Response Time
S10	Lost Carrier to Hang Up Delay
S11 *	DTMF Tone Duration
S12	Escape Code Guard Time
S13	Not Used
S14 *	Bit Mapped Option Register
S15	Not Used
S16	Modem Test Options
S17	Not Used
S18 *	Test Timer
S19	Not Used
S20	Not Used
S21 *	Bit Mapped Options Register
S22 *	Bit Mapped Options Register
S23 *	Bit Mapped Options Register
S24	Not Used
S25 *	Delay to DTR (Sync Only)
S26 *	RTS to CTS Delay (Half Dup.)
S27 *	Bit Mapped Options Register
S31 *	Bit Mapped Options Register

NOTE:
* These S registers can be stored in the NVRAM.

DIALING

Dial modifiers are available for adding conditions to dialed phone numbers.

Dial Modifiers

P	Pulse Dial
R	Originate call in Answer Mode
T	Tone Dial
S	Dial a stored number
W	Wait for dial tone
,	Delay a dial sequence
;	Return to command state
!	Initiate a flash
@	Wait for quiet

Example:

Terminal: AT &Z0 = T 1 (602) 555-1212
Modem: OK

Result: Modem stores the Tone Dial (T) modifier and phone number T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:

Terminal: AT DS0
Modem: T16025551212

Result: Modem dials phone number and attempts to establish a connection.

or by turning on $\overline{DTR}$ when in Synchronous Mode 2. Up to 33 symbols (dial digits and dial modifiers) may be stored. Spaces and other delimiters are ignored and do not need to be included in the count. If more than 33 symbols are supplied, the dial string will be truncated to 33.

POWER MANAGEMENT

The flexible power management controls allow for a variety of command and hardware driver options. The power down sequence is initiated by placing a logic "low" on pin 15 ($\overline{PD}$) of the 89C026LT. The laptop can control the $\overline{PD}$ signal directly. If such a signal is unavailable, $\overline{PD}$ can be controlled by communications software via $\overline{DTR}$. Lack of data activity or an in-coming ring signal can also be used to control $\overline{PD}$.

Placing the crystal on the 89C026LT (Figure 10) allows it to reduce power consumption by turning off the oscillator. When online and connected to a remote modem, the power consumption for the 89C024LT is typically 400 mW. Additionally, when the 89027 is not needed (on-hook, not connected to a remote modem) the 89C026LT places it in standby. In standby the chip set power consumption is typically 255 mW. When powered down via the $\overline{PD}$ pin on the 89C026LT, the chip set typically consumes 5 mW. Minimum memory-system power-consumption can be achieved by chip selecting memory only when addressed.

APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 4. The DAA section shown in this diagram may be implemented using the suggested diagram in Figure 5. Figure 6 shows the use of the power-down feature.

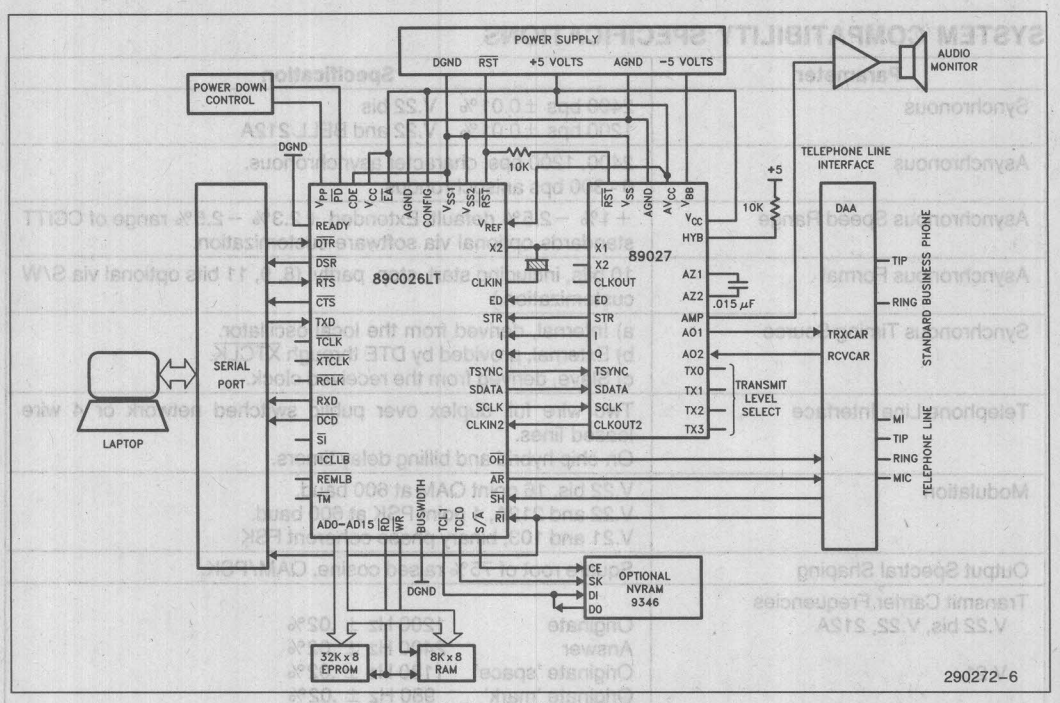


Figure 4. Typical Laptop Modem

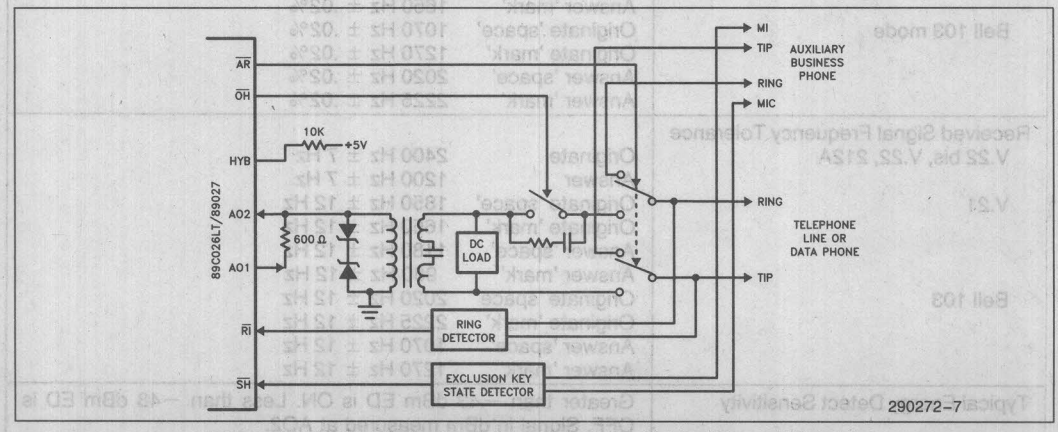


Figure 5. Typical Telephone Line Interface Using Internal Hybrid

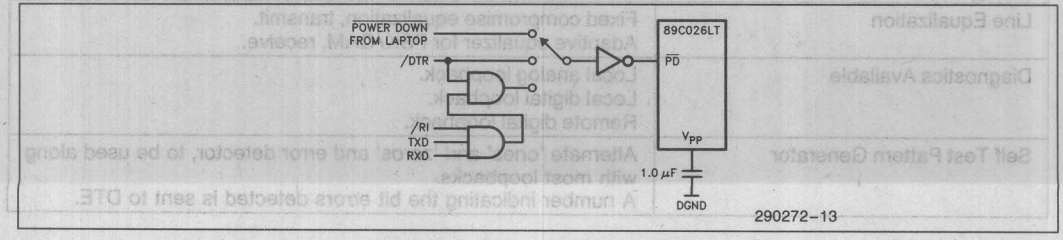


Figure 6. Power-Down Control

SYSTEM COMPATIBILITY SPECIFICATIONS

Parameter	Specification
Synchronous	2400 bps ±0.01% V.22 bis 1200 bps ±0.01% V.22 and BELL 212A
Asynchronous	2400, 1200 bps, character asynchronous. 0 - 300 bps anisochronous.
Asynchronous Speed Range	+1% −2.5% default. Extended +2.3% −2.5% range of CCITT standards optional via software customization.
Asynchronous Format	10 bits, including start, stop, parity. (8, 9, 11 bits optional via S/W customization.)
Synchronous Timing Source	a) Internal, derived from the local oscillator. b) External, provided by DTE through $\overline{\text{XTCLK}}$. c) Slave, derived from the received clock.
Telephone Line Interface	Two wire full duplex over public switched network or 4 wire leased lines. On-chip hybrid and billing delay timers.
Modulation	V.22 bis, 16 point QAM at 600 baud. V.22 and 212A, 4 point PSK at 600 baud. V.21 and 103, binary phase coherent FSK
Output Spectral Shaping	Square root of 75% raised cosine, QAM/PSK.
Transmit Carrier Frequencies V.22 bis, V.22, 212A V.21 Bell 103 mode	 Originate 1200 Hz ± .02% Answer 2400 Hz ± .02% Originate 'space' 1180 Hz ± .02% Originate 'mark' 980 Hz ± .02% Answer 'space' 1850 Hz ± .02% Answer 'mark' 1650 Hz ± .02% Originate 'space' 1070 Hz ± .02% Originate 'mark' 1270 Hz ± .02% Answer 'space' 2020 Hz ± .02% Answer 'mark' 2225 Hz ± .02%
Received Signal Frequency Tolerance V.22 bis, V.22, 212A V.21 Bell 103	 Originate 2400 Hz ± 7 Hz Answer 1200 Hz ± 7 Hz Originate 'space' 1850 Hz ± 12 Hz Originate 'mark' 1650 Hz ± 12 Hz Answer 'space' 1180 Hz ± 12 Hz Answer 'mark' 980 Hz ± 12 Hz Originate 'space' 2020 Hz ± 12 Hz Originate 'mark' 2225 Hz ± 12 Hz Answer 'space' 1070 Hz ± 12 Hz Answer 'mark' 1270 Hz ± 12 Hz
Typical Energy Detect Sensitivity	Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at AO2.
Energy Detect Hysteresis	A minimum Hysteresis of 2 dB for QAM scrambled mark.
Line Equalization	Fixed compromise equalization, transmit. Adaptive equalizer for PSK/QAM, receive.
Diagnostics Available	Local analog loopback. Local digital loopback. Remote digital loopback.
Self Test Pattern Generator	Alternate 'ones' and 'zeros' and error detector, to be used along with most loopbacks. A number indicating the bit errors detected is sent to DTE.

RECEIVER PERFORMANCE SPECIFICATIONS

Test Cases		Typical SNR for 10^{-5} BER Performance	
Data Mode	Rx Level (dBm)	Answer (dB)	Originate (dB)
V.22 bis Synchronous	−30	16	16.5
	−40	16.5	18
V.22/Bell 212A Synchronous	−30	6.5	6.5
	−40	6.5	6.5
V.21 Asynchronous	−30	9	7.5
	−40	9	8
Bell 103 Asynchronous	−30	10	11.5
	−40	10	11.5

Test Conditions:

— Receive Signal (Rx) measured at A02 (transmit level set at −9 dBm)

— Unconditioned 3002 Line

— 3 KHz Flat-Band Noise

PERFORMANCE SPECIFICATIONS

Parameter	Min	Typ	Max	Units	Comments
DTMF Level		4.0		dBm	at AO1
DTMF Second Harmonic			−35	dB	HYB enabled into 600Ω
DTMF Twist (Balance)		3		dB	
Default DTMF Duration		100		ms	Software Controlled
Pulse Dialing Rate		10/20		pps	Software Controlled
Pulse Dialing Make/Break		39/61 33/67		% %	US UK, Hong Kong
Pulse Interdigit Interval		785		ms	
Billing Delay Interval			2.1	sec	
Guard Tone Frequency Amplitude Frequency Amplitude		540 −3 1800 −6		Hz dB Hz dB	referenced to High Channel transmit. QAM/PSK Modes Only
Dial Tone Detect Duration		3.0		sec	
Ringback Tone Detect Duration Cadence		 0.75 1.5		 sec	 Off/On Ratio
Busy Tone Detect Duration Cadence	 0.67	 0.2	 1.5	 sec	 Off/On Ratio

4

89C026LT OVERVIEW

The 89C026LT processor performs data manipulation, signal processing and user interface functions. It requires a single ROM and RAM to execute standard, and/or custom code with high level protocol functions. A block diagram of the 89C026LT is provided in Figure 7.

89C026LT contains a TTL compatible serial link to DTE equipment, along with a full complement of V.24/RS-232-C control signals. A UART or USART may be used to transfer data to and from a microcomputer bus. The 89C026LT supports the industry standard AT command set facilitating compatability with most PC software.

During transmit operation, the 89C026LT synthesizes DTMF tones and the 300 BPS FSK modem signal and transmits them to the 89027 as digitized amplitude samples. During 1200 and 2400 BPS operation, DPSK and QAM is used to send 2 to 4 bits of information respectively at 600 baud to the AFE. Because the QAM coding technique is an inherently synchronous transmission mechanism, in the case of asynchronous QAM transmission, the asynchronous data is synchronized by adding or deleting stop bits. Following the synchronization process, the 89C026LT transmits digitized phase and amplitude samples to 89027 over the high speed serial link.

In the receive operation, the information is received by the 89C026LT from the 89027 as two signals which are 90 degrees phase shifted from each other. These analog signals are then digitized by the A/D converter resident on the 89C026LT. By using DSP algorithms, the received signals are processed using adaptive equalization for telephone line delay, amplitude distortion and gain adjustments and the signal demodulated. Following demodulation, the data is unscrambled, and if necessary, returned to asynchronous format.

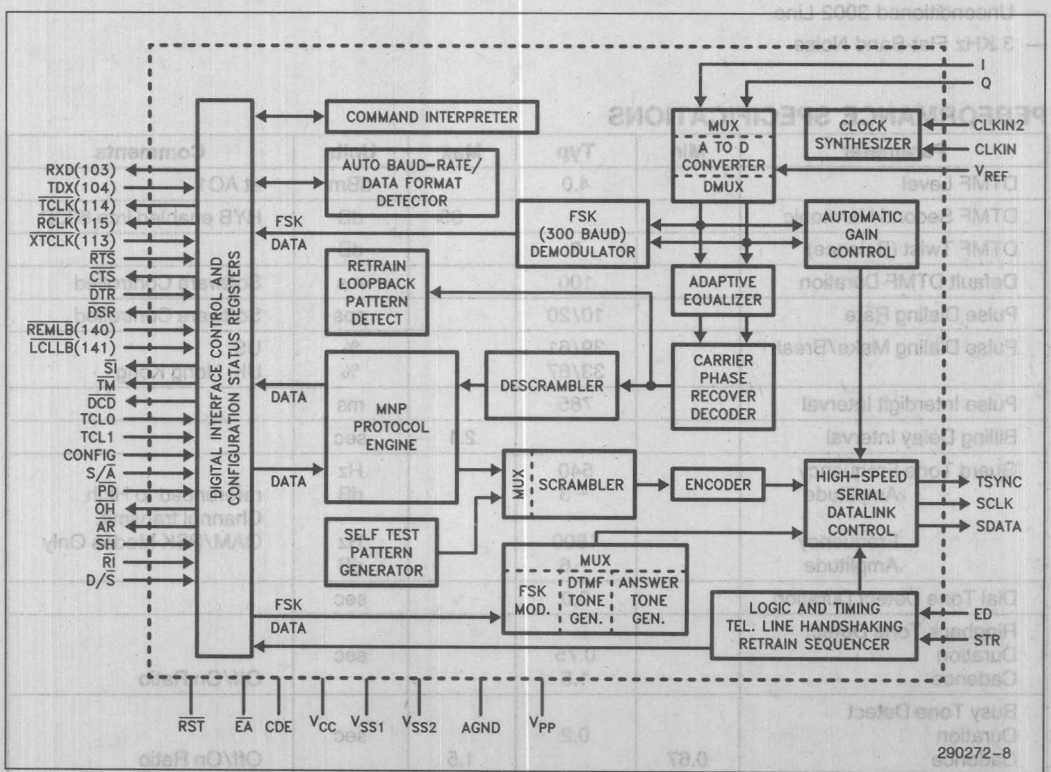


Figure 7. 89C026LT Block Diagram

89C026LT PINOUT

Symbol	Function (89C024LT)	Direction(4)	Pin No.
CLKIN	12.96 MHz master clock from 89027	In	67
CLKIN2	270 KHz from 89027	In	44
$\overline{RST}$	Chip reset (active low)	In	16
I	In-phase received signal	In	11
Q	Quadrature-phase received signal	In	10
STR	Symbol Timing from 89027	In	24
ED	Energy Detect input	In	9
TSYNC	Transmitter sync pulse to 89027	Out	35
SDATA	Serial Data to 89027	Out	17
SCLK	Serial Clock to 89027	Out	18
$\overline{OH}$	Off-Hook control to DAA	Out	33
$\overline{SH}$	Switch-Hook from dataphone	In	5
$\overline{RI}$	Ring Indicator from DAA	In	42
$\overline{AR}$	Aux Relay control to DAA	Out	38
TCL1	NVRAM Data I/O	I/O	20
TCL0	NVRAM CLK	Out	19
$\overline{PD}$	Power-down control	In	15
S/$\overline{A}$	NVRAM CE	Out	21
D/$\overline{S}$	Dumb/Smart mode select	In	6
CONFIG	Reserved for future use (V_{CC})(2)	In	8
$\overline{TM}$	Test Mode Indicator	Out	39
TXD	Transmitted data from DTE	In	27
RXD	Received data to DTE	Out	29
$\overline{RTS}$	Request to send from DTE	In	22
$\overline{CTS}$	Clear to Send to DTE	Out	23
$\overline{DSR}$	Data Set Ready to DTE	Out	30
$\overline{DCD}$	Data Carrier Detect to DTE	Out	31
$\overline{DTR}$	Data Terminal Ready from DTE	In	25
$\overline{RCLK}$	Received clock to DTE	Out	34
$\overline{TCLK}$	Transmit clock to DTE	Out	28
$\overline{XTCLK}$	External timing clock from DTE	In	26
$\overline{SI}$	Speed Indicator to DTE	Out	32
$\overline{REMLB}$	Remote Loopback Command from DTE	In	7
$\overline{LCLLB}$	Local Loopback Command from DTE	In	4
V_{CC}	Positive power supply (+5V)	+5V	1
CDE	Clock detect enable (V_{SS})(1)	GND	14
V_{REF}	A/D converter reference	+5V	13
V_{SS1}	Digital ground	GND	36
V_{SS2}	Digital ground	GND	68
AGND	Analog ground	AGND	12
V_{PP}	Timing pin for return from power-down	In	37
$\overline{EA}$	External Memory enable	In	2
AD0-AD15	External memory access address/data(3)	I/O	60–45
$\overline{AA}$	Auto Answer(3)	Out	60
$\overline{JS}$	Jack Select(3)	Out	59
$\overline{CD}$	Carrier Detect Indicator(3)	Out	58
$\overline{MR}$	Modem Ready Indicator(3)	Out	57
$\overline{REL}$	MNP Reliable Link Active(3)	Out	56
$\overline{CLASS5}$	MNP Class 5 Compression Active(3)	Out	55
$\overline{ERR}$	Error detected by MNP(3)	Out	54

4

89C026LT PINOUT (Continued)

Symbol	Function (89C026LT)	Direction(4)	Pin No.
NMI	Non-maskable Interrupt(V_{SS})(1)	In	3
X2	Crystal output	Out	66
CLKOUT	Clk output	Out	65
BUSWIDTH	Bus Width	In	64
INST	External memory instruction fetch	Out	63
ALE	Address latch enable	Out	62
$\overline{RD}$	External memory read	Out	61
READY	External memory ready	In	43
$\overline{BHE}$	External memory bus high enable	Out	41
$\overline{WR}$	External memory write	Out	40

NOTES:
1. Pins marked with (V_{SS}) must be connected to V_{SS}.
2. Pins marked with (V_{CC}) must be connected to V_{CC}.
3. AD0–AD3 are used as $\overline{AA}$, $\overline{JS}$, $\overline{CD}$, $\overline{MR}$, $\overline{REL}$, $\overline{CLASS\ 5}$, and $\overline{ERR}$ respectively.
4. Pins with direction "IN" must not be left floating.

89C026LT PIN DESCRIPTION

$\overline{XTCLK}$

Transmitter timing from DTE, when external clock option is selected.

TXD

The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89C026LT samples this data on the rising edges of $\overline{TCLK}$.

$\overline{TCLK}$

Clock output from 89C026LT as timing source for data exchange from DTE to modem. Serial data is read on the rising edges of the $\overline{TCLK}$. This output is High in asynchronous mode.

RXD

The serial data to DTE. A logic 'high' is mark. In synchronous mode, the rising edge of $\overline{RCLK}$ occurs in the middle of RXD.

$\overline{RCLK}$

Synchronous clock output. Rising edge of $\overline{RCLK}$ occurs in the middle of each RXD bit. This pin remains High in asynchronous mode.

$\overline{PD}$

Power-down control. A low on this input pin, in conjunction with the +En and +Tn commands, will cause the modem to go into a power-down mode.

V_{PP}

Timing pin for return from power-down. Connect a 1.0 μf capacitor between V_{PP} and V_{SS} if the power-down option is used. This capacitor causes an internal timing circuit to give the oscillator time to stabilize before turning on internal clocks. This pin may be left floating or connected through a 1.0 μF capacitor to V_{SS} if power-down mode is not required.

$\overline{TM}$

A low indicates maintenance condition in the modem.

$\overline{DCD}$

In async operation, $\overline{DCD}$ remains low regardless of data carrier (default), or it can be programmed to indicate received carrier signal is within the required timing and amplitude limits. In sync operation low indicates the received carrier signal is within the required timing and amplitude limits.

$\overline{DSR}$

A low indicates modem is off-hook, is in data transmission mode, and the answer tone is being exchanged. $\overline{CTS}$ low indicates modem is prepared to accept data.

$\overline{RTS}$

In async mode $\overline{RTS}$ is ignored. Under command control, in sync mode $\overline{RTS}$ can be ignored, or the modem can respond with a Low on $\overline{CTS}$.

$\overline{DTR}$

&D0 command will cause the modem to ignore $\overline{DTR}$. For &D1 the modem assumes the asynchronous command state on a low-to-high transition of the $\overline{DTR}$ circuit. The &D2 command does the same as &D1 except the state of $\overline{DTR}$ will enable/disable auto answer. A low-to-high transition of $\overline{DTR}$ after the &D3 command will cause the modem to assume the initialization state.

TCL1, TCL0

These pins are used as the serial clock and data for interface to an NVRAM. Refer to Figure 3. TCL0 is used to output a clock and serial data is transferred in on TCL1.

$\overline{AR}$

This Auxiliary relay control is for switching a relay for voice or data calls. High is voice, low is data.

$\overline{RI}$

A low signal from DAA indicates line ringing. This input is ignored when the modem is configured for leased line. This signal should follow the ring cadence.

$\overline{OH}$

Low sets an off hook condition, high indicates an on hook. When dialing, this signal is used to pulse dial the line.

$\overline{SH}$

Used as a telephone voice to data switch or vice versa. Any logic level transition will toggle the modem state between voice and data.

$\overline{AA}$

Used as an indicator for Auto Answer status and Ring indicator. Active low.

$\overline{LCLLB}$

A low will set the modem in the local analog loopback test mode. Logic Low levels applied simultaneously to $\overline{REMLB}$ and $\overline{LCLLB}$ pins, sets the modem to the local digital loopback.

$\overline{REMLB}$

A low on this pin initiates a remote loopback condition.

$\overline{CD}$

A Low indicates the presence of carrier signal on the line.

$\overline{MR}$

A low indicates the presence of the DSR signal. Toggling indicates that a test mode is active.

$\overline{REL}$

A low indicates that an MNP reliable link has been established.

$\overline{CLASS5}$

A low indicates that MNP Class 5 (data compression) is in operation.

$\overline{ERR}$

Goes low for 1 second whenever MNP detects an error.

$\overline{SI}$

Selects one of the two data rates or ranges of rates in the DTE to correspond to the rate in modem. Low selects the higher rate (2400 CCITT/1200 BELL) or range of rates. High selects the Low rate or range of rates.

D/$\overline{S}$

A low on this pin will indicate the smart mode which will respond to all commands. A High will ignore all commands.

V_{REF}

Voltage reference for the analog to digital converter should be connected to the 89027 AVcc.

CDE

This pin must be connected to V_{SS}.

S/$\overline{A}$

The function of this pin is re-defined as external NVRAM CE.

CONFIG

Reserved for future use. This signal should be pulled high.

$\overline{EA}$

When high, memory access from address 2000H to 4000H are directed to on-chip ROM. When low, all Memory access is directed to off-chip memory. This pin must be tied high.

$\overline{JS}$

Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

BUSWIDTH

When high, external memory accesses are 16 bits wide. When low, external memory accesses are 8 bits wide. This pin must be tied low.

READY

When high, no wait states are inserted in external memory accesses. When low, one wait state is inserted in each external memory access.

4

89C026LT ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70°C
Storage Temperature . . . . . −40°C to +125°C
Voltage from Any Pin to V_{SS} or AGND . . . . . −0.5V to +7.0V
Average Output Current from Any Pin . . . . . 10 mA
Power Dissipation . . . . . 1.5 Watts

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T_A	Ambient Temperature Under Bias	0	+70	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V
V_{REF}	Analog Supply Voltage	4.75	5.25	V
f_{OSC}	CLKIN Frequency	12.95870	12.96130	MHz

NOTE:
The AGND and V_{SS} on both the 89C026LT and the 89027 must be nominally at the same potential.

D.C. CHARACTERISTICS

Symbol	Parameter	Min	Typ(7)	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5		+0.8	V	
V_{IH}	Input High Voltage(1)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage on CLKIN	0.7 V_{CC}		V_{CC} + 0.5	V	
V_{IH2}	Input High Voltage on RESET	2.2		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage			0.3 0.45 1.5	V V V	I_{OL} = 200 μA I_{OL} = 3.2 mA I_{OL} = 7 mA
V_{OH}	Output High Voltage(4)	V_{CC} − 0.3 V_{CC} − 0.7 V_{CC} − 1.5			V V V	I_{OH} = −200 μA I_{OH} = −3.2 mA I_{OH} = −7 mA
V_{OH1}	Output High Voltage(3)	V_{CC} − 0.3 V_{CC} − 0.7 V_{CC} − 1.5			V V V	I_{OH} = −10 μA I_{OH} = −30 μA I_{OH} = −60 μA
I_{LI}	Input Leakage Current(5)			±10	μA	0 < V_{IN} < V_{CC} − 0.3V
I_{LI1}	Input Leakage Current(6)			±3	μA	0 < V_{IN} < V_{REF}
I_{IL}	Logical 0 Input Current(3)			−50	μA	V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current in RESET(2) (ALE, $\overline{RD}$, $\overline{WR}$, $\overline{BHE}$, INST, SCLK)			−850	μA	V_{IN} = 0.45V

D.C. CHARACTERISTICS (Continued)

Symbol	Parameter	Min	Typ(7)	Max	Units	Test Conditions
I_{REF}	A/D Converter Reference Current		2	5	mA	CLKIN = 12.96 MHz $V_{CC} = V_{PP} = V_{REF} = 5.25$
I_{CC1}	Active Mode Current (Typical)		45	60	mA	CLKIN = 12.96 MHz
R_{RST}	RESET Pullup Resistor	6K		50K	Ω	
C_S	Pin Capacitance (Any Pin to V_{SS})			10	pF	f_{TEST} = 1.0 MHz
I_{PD}	Power-Down Mode Current		5	50	μA	$V_{CC} = V_{PP} = V_{REF} = 5.25$

NOTES:
(Notes apply to all specifications)
1. All pins except RESET and CLKIN.
2. Holding these pins below V_{IH} in RESET may cause the part to enter test modes.
3. TCL0, TCL1, S/$\overline{A}$, RTS, $\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, $\overline{SI}$, $\overline{OH}$.
4. $\overline{BHE}$, INST, CLKOUT, RESET, $\overline{TCLK}$, RXD, $\overline{RCLK}$, TSYNC, $\overline{TM}$, SCLK, SDATA. The V_{OH} specification is not valid for RESET.
5. CDE, EA, READY, BUSWIDTH, NMI, STR, DTR, XTCLK, TXD, B/C, CLKIN2, and RI.
6. S/D, $\overline{SH}$, $\overline{REMLB}$, $\overline{LCLLB}$, I, Q, CONFIG, ED.
7. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature and $V_{REF} = V_{CC} = 5V$.

A.C. CHARACTERISTICS (Over specified operating conditions)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, f_{OSC} 12.96 MHz

This system must meet these specifications to work with 89C026LT:

Symbol	Parameter	Min	Max	Units	Notes
T_{AVYV}	Address Valid to READY Setup		$2T_{OSC} - 85$	ns	
T_{LLYV}	ALE Low to READY Setup		$T_{OSC} - 72$	ns	
T_{YLYH}	Non READY Time	No Upper Limit		ns	
T_{CLYX}	READY Hold after CLKOUT Low	0	$T_{OSC} - 30$	ns	(Note 1)
T_{LLYX}	READY Hold after ALE Low	$T_{OSC} - 15$	$2T_{OSC} - 40$	ns	(Note 1)
T_{AVGV}	Address Valid to Buswidth Setup		$2T_{OSC} - 85$	ns	
T_{LLGV}	ALE Low to Buswidth Setup		$T_{OSC} - 70$	ns	
T_{CLGX}	Buswidth Hold after CLKOUT Low	0		ns	
T_{AVDV}	Address Valid to Input Data Valid		$3T_{OSC} - 67$	ns	
T_{RLDV}	$\overline{RD}$ Active to Input Data Valid		$T_{OSC} - 23$	ns	
T_{CLDV}	CLKOUT Low to Input Data Valid		$T_{OSC} - 50$	ns	
T_{RHDZ}	End of $\overline{RD}$ to Input Data Float		$T_{OSC} - 20$	ns	
T_{RXDX}	Data Hold after $\overline{RD}$ Inactive	0		ns	

NOTE:
1. If max is exceeded, additional wait states will occur.

A.C. CHARACTERISTICS (Continued)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall tines = 10 ns, f_{OSC} 12.96 MHz

The 89C026LT will meet these specifications:

Symbol	Parameter	Min	Max	Units	Notes
F_{CLKIN}	Oscillator Frequency	12.95870	12.96130	MHz	
T_{OSC}	Oscillator Period	$1/F_{CLKIN(MAX)}$	$1/F_{CLKIN(MIN)}$	ns	
T_{XHCH}	F_{CLKIN} High to CLKOUT High or Low	40	110	ns	(Note 1)
T_{CLCL}	CLKOUT Cycle Time	$2\,T_{OSC}$		ns	
T_{CHCL}	CLKOUT High Period	$T_{OSC} - 10$	$T_{OSC} + 10$	ns	
T_{CLLH}	CLKOUT Falling Edge to ALE Rising	−5	15	ns	
T_{LLCH}	ALE Falling Edge to CLKOUT Rising	−15	15	ns	
T_{LHLH}	ALE Cycle Time	$4\,T_{OSC}$		ns	
T_{LHLL}	ALE High Period	$T_{OSC} - 10$	$T_{OSC} + 10$	ns	
T_{AVLL}	Address Setup to ALE Falling Edge	$T_{OSC} - 20$		ns	
T_{LLAX}	Address Hold after ALE Falling Edge	$T_{OSC} - 40$		ns	
T_{LLRL}	ALE Falling Edge to $\overline{RD}$ Falling Edge	$T_{OSC} - 40$		ns	
T_{RLCL}	$\overline{RD}$ Low to CLKOUT Falling Edge	5	30	ns	
T_{RLRH}	$\overline{RD}$ Low Period	$T_{OSC} - 5$	$T_{OSC} + 25$	ns	
T_{RHLH}	$\overline{RD}$ Rising Edge to ALE Rising Edge	T_{OSC}	$T_{OSC} + 25$	ns	(Note 2)
T_{RLAZ}	$\overline{RD}$ Low to Address Float		10	ns	
T_{LLWL}	ALE Falling Edge to $\overline{WR}$ Falling Edge	$T_{OSC} - 10$		ns	
T_{CLWL}	CLKOUT Low to $\overline{WR}$ Falling Edge	0	25	ns	
T_{QVWH}	Data Stable to $\overline{WR}$ Rising Edge	$T_{OSC} - 23$		ns	
T_{CHWH}	CLKOUT High to $\overline{WR}$ Rising Edge	−10	10	ns	
T_{WLWH}	$\overline{WR}$ Low Period	$T_{OSC} - 30$	$T_{OSC} + 5$	ns	
T_{WHQX}	Data Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{WHLH}	$\overline{WR}$ Rising Edge to ALE Rising Edge	$T_{OSC} - 10$	$T_{OSC} + 15$	ns	(Note 2)
T_{WHBX}	$\overline{BHE}$, INST, Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{RHBX}	$\overline{BHE}$, INST, Hold after $\overline{RD}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{WHAX}	AD8–15 Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 50$		ns	
T_{RHAX}	AD8–15 Hold after $\overline{RD}$ Rising Edge	$T_{OSC} - 25$		ns	

NOTES:
1. Typical specifications, not guaranteed.
2. Assuming back-to-back bus cycles.

WAVEFORMS

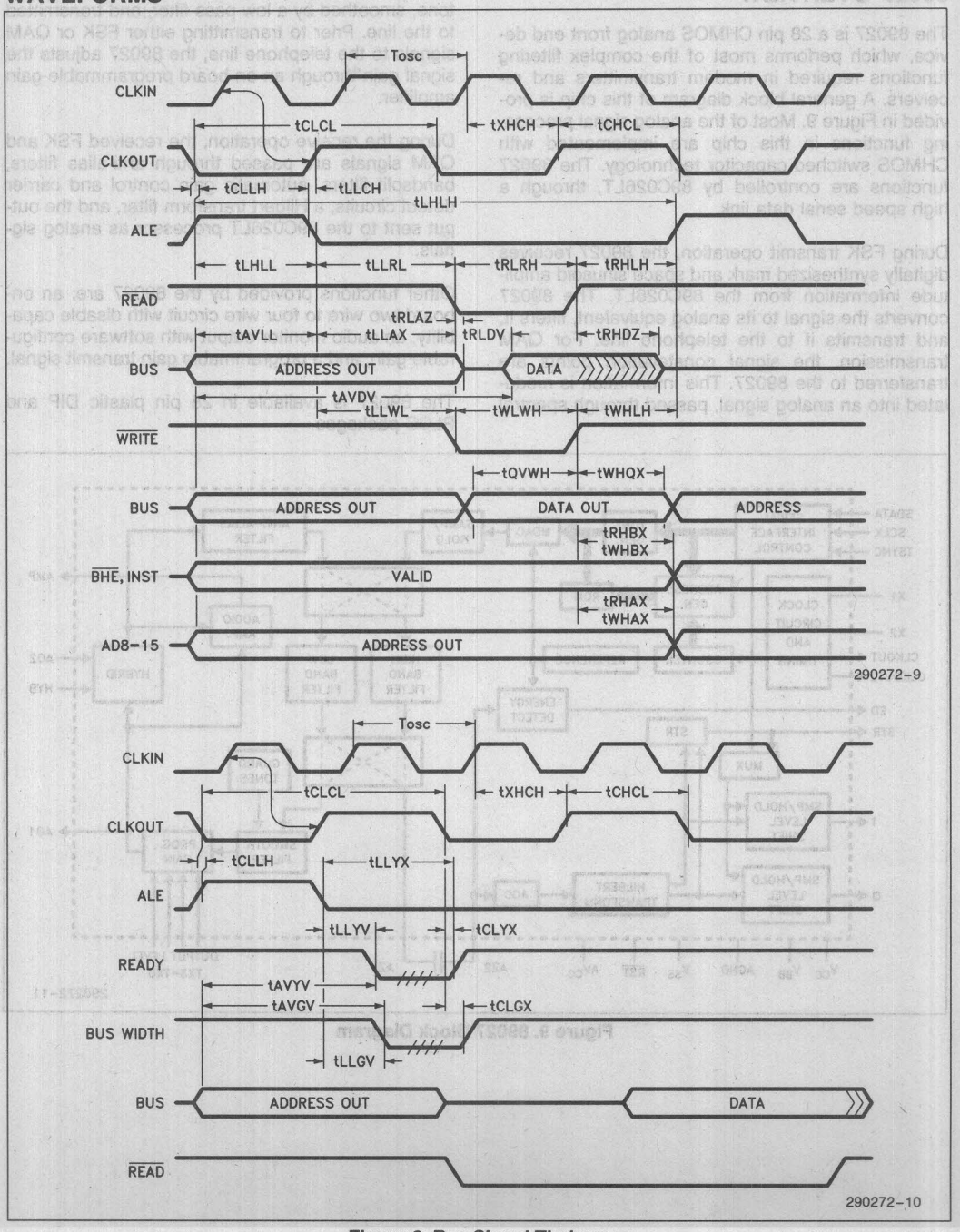


Figure 8. Bus Signal Timings

89027 OVERVIEW

The 89027 is a 28 pin CHMOS analog front end device, which performs most of the complex filtering functions required in modem transmitters and receivers. A general block diagram of this chip is provided in Figure 9. Most of the analog signal processing functions in this chip are implemented with CHMOS switched capacitor technology. The 89027 functions are controlled by 89C026LT, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89C026LT. The 89027 converts the signal to its analog equivalent, filters it, and transmits it to the telephone line. For QAM transmission, the signal constellation points are transferred to the 89027. This information is modulated into an analog signal, passed through spectral shaping filters, combined with the necessary guard tone, smoothed by a low pass filter, and transmitted to the line. Prior to transmitting either FSK or QAM signals to the telephone line, the 89027 adjusts the signal gain through an on-board programmable gain amplifier.

During the receive operation, the received FSK and QAM signals are passed through anti-alias filters, bandsplit filters, automatic gain control and carrier detect circuits, a Hilbert transform filter, and the output sent to the 89C026LT processor as analog signals.

Other functions provided by the 89027 are: an on-board two wire to four wire circuit with disable capability, an audio monitor output with software configurable gain, and a programmable gain transmit signal.

The 89027 is available in 28 pin plastic DIP and PLCC packages.

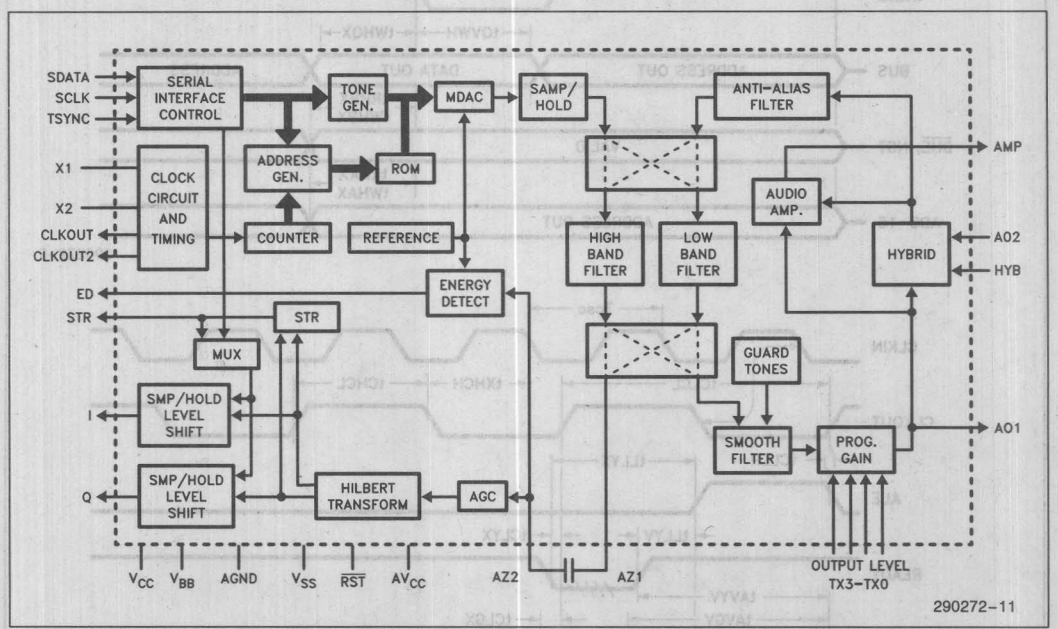


Figure 9. 89027 Block Diagram

89027 PINOUT

Symbol	Function (89027)	Direction	Pin No.
V_{CC}	Positive Power Supply (Digital)	+5V	28
V_{BB}	Negative Power Supply	−5V	15
V_{SS}	Digital Ground	DGND	24
AGND	Analog Ground	AGND	21
AV_{CC}	Positive Power Supply (Analog)	+5	7
X1	Xtal Oscillator	In	23
X2	Xtal Oscillator	Out	25
CLKOUT	12.96 MHz Clock Output to 89C026LT	Out	26
CLKOUT2	270 KHz Clock Output to 89C026LT	Out	19
$\overline{RST}$	Chip reset (active low)(3)	In	20
HYB	Enable on-chip hybrid(1)	In	10
AZ1	Auto-zero capacitor	Out	16
AZ2	Auto-zero capacitor	In	17
SDATA	Serial data from 89C026LT	In	2
SCLK	Serial clock from 89C026LT	In	1
TSYNC	Transmitter sync from 89C026LT	In	3
STR	Symbol timing to 89C026LT	Out	27
ED	Receiver energy detect to 89C026LT	Out	18
I	In phase received signal to 89C026LT	Out	13
Q	Quadrature-phase received signal to 89C026LT	Out	14
AO1	Transmitter output	Out	6
AO2	Receiver input	In	12
AMP	Output to monitor speaker	Out	11
TX0	Transmitter level control (LSB)(1)	In	9
TX1	Transmitter level control(1)	In	8
TX2	Transmitter level control(1)	In	5
TX3	Transmitter level control (MSB)(1)	In	4
NC	(Note 2)	In	22

NOTE:
1. When held high, these pins must be connected through 10K resistors to V_{CC}.
2. Reserved Pin. Must be left No Connect.
3. Connect to reset circuitry through a 10K resistor.

89027 Pinout Description

TX0-3

These four pins control the transmitted signal level. Refer to Transmit Level Table.

HYB

This pin enables the on-chip hybrid. A line impedance matching network must be connected between AO1 and AO2 when HYB is enabled. If HYB is disabled and an external 4W/2W hybrid is used, the hybrid receive path must be amplified by 6 dB.

AO1

Transmitter output.

AO2

Receiver input.

AMP

This output can be used to monitor the call progress tones and operation of the line.

ABSOLUTE MAXIMUM RATINGS(2)

Temperature Under Bias .....0 to +70° C
Storage Temperature .....−40 to +125° C
All Input and Output Voltages with Respect to V_{BB} .....−0.3V to +13.0V
All Input and Output Voltages with Respect to V_{CC} & AV_{CC} .....−13.0V to 0.3V
Power Dissipation 1.35W
Voltage with Respect to V_{SS}(1) .....−0.3V to 6.5V

NOTES:
1. Applies to pins SCLK, SDATA, TSYNC, $\overline{RST}$, HYB, TX0–TX3 only.
2. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T_A	Ambient Temperature Under Bias	0	+70	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V
V_{BB}	Analog Supply Voltage	−4.75	−5.25	V

POWER DISSIPATION Ambient Temp = 0° to 70° C, V_{CC} = AV_{CC} = 5 ± 5%, V_{SS} = AGND = 0V.

Symbol	Parameter	Min	Typ	Max	Units
$Aicc_1$	AV_{CC} Operating Current		15	21	mA
Icc_1	V_{CC} Operating Current		5	6	mA
Ibb_1	V_{BB} Operating Current		−15	−21	mA
$Aicc_s$	AV_{CC} Standby Current		0.2	1	mA
Icc_s	V_{CC} Standby Current		5	6	mA
Ibb_s	V_{BB} Standby Current		−0.6	−2	mA
$Aicc_p$	AV_{CC} Power-Down Current		100		μA
Icc_p	V_{CC} Power-Down Current		450		μA
Ibb_p	V_{BB} Power-Down Current		450		μA
Pdo	Operating Power Dissipation		175	250	mW
Pds	Standby Power Dissipation		30	50	mW
Pd_p	Power Down Power Dissipation		5		mW

D.C. CHARACTERISTICS (T_A=0°C to 70°C, AV_{CC} = V_{CC} = 5V ±5%, V_{BB} = 5V ±5%, AGND = V_{SS} = 0V), supply voltage must be at the same potential as the 89C026LT power supply. Typical Values are for T_A = 25°C and nominal power supply values. V_{CC}, and AV_{CC}. V_{CC}, AV_{CC} and 89C026LT V_{REF} must be nominally at the same potential.

Inputs: TX0, TX1, TX2, TX3, HYB, $\overline{RST}$
Outputs: CLKOUT

Symbol	Parameter	Min	Max	Units	Test Conditions
Iil	Input Leakage Current	−10	+10	μA	$V_{SS} \le Vin \le V_{CC}$
Vil	Input Low Voltage	V_{SS}	0.8	V	
Vih	Input High Voltage	2.0	V_{CC}	V	
Vol	Output Low Voltage		0.4	V	Iol ≥ −1.6mA,1 TTL load
Voh	Output High Voltage	2.4		V	Ioh ≤ 50μa, 1 TTL load
Vcol	CLKOUT Low Voltage		0.4	V	Load Capacitance = 60 pF
Vcoh	CLKOUT High Voltage	0.7 V_{CC}		V	Load Capacitance = 60 pF

A.C. CHARACTERISTICS (T_A = 25°C, V_{CC} = AV_{CC} = 5V, V_{SS} = AGND = 0V, V_{BB} = −5V)

ANALOG INPUTS: AO2

Parameter	Min	Typ	Max	Units	Test Condition
AO2 Receive Signal Level			−9	dBm	Hybrid Enabled
AO2 Input Resistance		10		MOhms	−2.5V<Vin<+2.5V
AO2 Allowed DC offset	−30		+30	mV	Relative to AGND

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF
Tolerance = ±20%
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.

4

CRYSTAL REQUIREMENTS(1)

Parameter	Min	Typ	Max	Units	Comments
Frequency Accuracy (0°C–70°C)	−0.0035%	12.96	+0.0035%	MHz	Refer to Figure 10
Rx		10	16	Ohms	
Cx		0.024		pF	
Co	5.1	5.6	6.1	pF	
C_L(2)	−5%	47	+5%	pF	2 Load Capacitors

NOTES:
1. Crystal Type: Parallel Resonant
2. Crystal manufacturers usually specify the accuracy of a parallel resonant circuit at a given "load capacitance". This "load capacitance" is specified to the crystal manufacturer as 33 pf. 33 pf includes the parallel combination of the capacitances seen at the pins of the crystal. These capacitances include C_L, IC pin capacitances, and a 3 ± 2 pf trace capacitance.

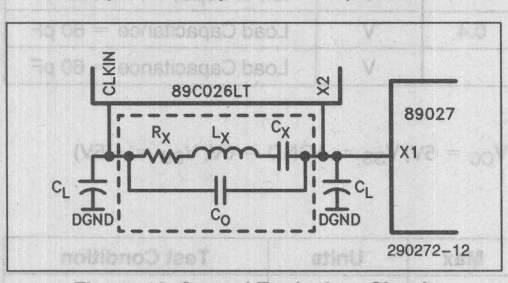


Figure 10. Crystal Equivalent Circuit

ANALOG OUTPUTS: A01, AMP

Parameter	Min	Typ	Max	Units	Comments
Load Resistance					
AO1	600			Ohms	
AMP	10			kOhms	
Load Capacitance					
AMP			100	pF	
Audio Amp Gain		−9		dB	Max
AO1 to Amp		−18		dB	Mid (Software Selectable)
		−26		dB	Min (Software Selectable)
		−70		dB	Off
Audio Amp Gain(1)		+12		dB	Max
AO2 to Amp		+3		dB	Mid (Software Selectable)
		−4		dB	Min (Software Selectable)
		−60		dB	Off

NOTE:
1. Assumes on-chip hybrid is enabled. When on-chip hybrid is disabled, gain with respect to AO2 is reduced by 6 dB.

Transmit Output Level(1)		
TX 3,2,1,0	Typ	Units
0 0 0 0	+5	dBm
0 0 0 1	+4	dBm
•	•	•
•	•	•
•	•	•
1 1 1 0	−9	dBm
1 1 1 1	−10	dBm

NOTE:
1. For PSK and QAM transmit signal. For FSK transmit signal levels, they are typically 1 dB lower. All signals are measured at A01. The tolerance for the transmit levels is ±1 dBm.

89C024XE CHIP SET USERS:

Note the following when using the 89C024LT Laptop chip set in place of the 89C024XE:

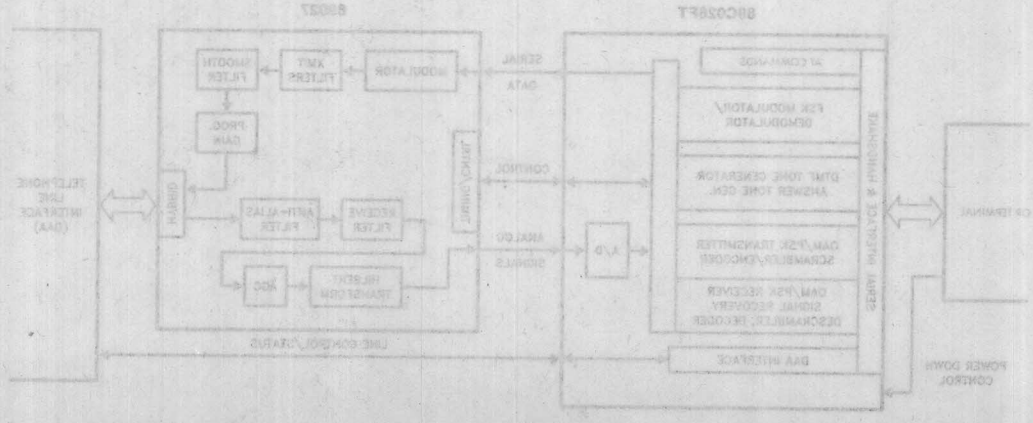
1. $\overline{PD}$ (B/$\overline{C}$ on 89C024XE) can be left tied to digital V_{CC} or can be connected to the power-down control. If the power down mode is used, a 1.0 μf capacitor must be connected from V_{PP} on the 89C026LT to digital ground.
2. Buswidth pin is tied directly to ground because 89C024LT uses 8-bit EPROM.

REFERENCE MANUALS

The Modem Reference Manual (Order Number 296235-002) contains pin descriptions, AT and MNP command descriptions, schematics, and important design guidelines for the 89C024LT, 89C024XE, and 89024 modem chip sets. The Modem Software Reference Manual (Order Number 296503-001) provides information about the modem software routines. Contact your local Intel sales office for the latest information.

3. No 74HC373 latch is required for AD8–AD15 because 89C026LT latches them internally.
4. A single chip select inverter is required from AD15 to select between EPROM and RAM memory.
5. Optional inverter from AD14 can be used to power-down RAM.
6. If minimum power-down currents are not required, the 89C024XE crystal configuration and specifications may be used.

4



89C024FT
V.42/42bis MODEM CHIP-SET

- CHMOS for Low Operating Power
- Low Standby Power
- Minimum Chip Count for Small size
- V.42 Compliant Error Correction (LAPM and *MNP4)
- V.42bis and MNP5 Data Compression Increase throughput up to 4 times with DTE rates of 9600
- AT Command Set
- Automatically Detects Remote Modem Type and Data Rate
- On-Chip Hybrid
- DTMF and Pulse Dialing
- On-Chip Serial Port and Handshake Signals for RS-232/V.24 Interface
- Serial Interface to External NVRAM
- Automatic Speed Matching in Reliable and Normal Modes
- Hardware and Software Flow Control
- Analog/Digital Loopback Diagnostics
- Synchronous Modes
- Easily Customized Command Set and Features
- Intel's V.42/42bis and MNP Software Co-Developed with R. Scott Associates**

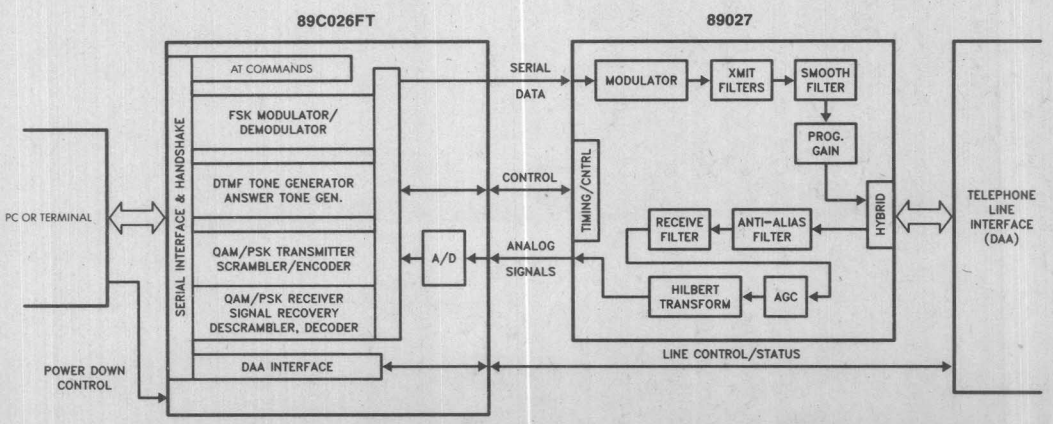


Figure 1. 89C024FT System Block Diagram

*MNP is a registered trademark of Microcom, Inc.
**R. Scott Associates, Inc., 5711 Six Forks Road, Suite 301, Raleigh, North Carolina 27609, (919) 846-7171

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied. Information contained herein supersedes previously published specifications on these devices from Intel. September 1990
© INTEL CORPORATION, 1990

Order Number: 290398-001

GENERAL DESCRIPTION

The Intel 89C024FT is a highly integrated, low power V.42/42bis compliant modem chip-set. This two chip solution is composed of the 89027 analog front end and the 89C026FT (an 80C196 based microcontroller).

V.42/42bis compliancy assures adherence to an international error correction (V.42: LAPM & MNP class 4) and data compression (V.42bis: BTLZ & MNP class 5) standards. V.42bis uses BTLZ (British Telecom Lempel Ziv) data compression algorithm to achieve throughputs of up to 4 times the transmission rate, effectively providing up to 9600 bps throughput with a 2400 bps modem.

The benefits of V.42/42bis compliancy are, compatibility with the installed base of MNP class 4 modems, and an increased throughput of up to 4:1. The chip-set also provides MNP class 5 operation. This provides 2:1 compression with the large installed base of MNP class 5 modems. These benefits allow the 89C024FT chip-set to provide fast and reliable data transfer with the current and upcoming installed base of modems products.

The 89C024FT chip-set, along with a Data Access Arrangement (DAA), a single 64K x 8 EPROM, a 32K x 8 static RAM, represent the circuitry necessary for implementing an auto dial/answer, 300 to 2400 bps, V.42/42bis compliant modem. Refer to figure 2 for a block diagram of this application. The system is compatible with the following CCITT and BELL transmission standards:

CCITT V.22bis
  2400 bps sync and async
  1200 bps sync and async
CCITT V.22 A & B
  1200 bps sync and async
CCITT V.21
  0 to 300 bps anisochronous
BELL 212A
  1200 bps sync and async
  300 bps fall-back mode
BELL 103
  0 to 300 bps anisochronous

This chip-set supports power-down modes that are selected via the AT command set, providing flexible power-down management control. The power-down modes make the 89C024FT a good fit for laptop computer applications. Power consumption for the chip-set is typically 400 mW during a connection. When powered-down, the chip-set consumes 5 mW.

A complete set of industry standard AT commands are provided for modem configuration and user interface. Additional commands have been implemented for power down modes and V.42/42bis/MNP feature control. Virtually all PC software written for the AT command set can also be used with this chip-set. Alternatively, in applications where user proprietary modem control commands and features are desired, the user can replace the 89C024FT command module with custom proprietary software.

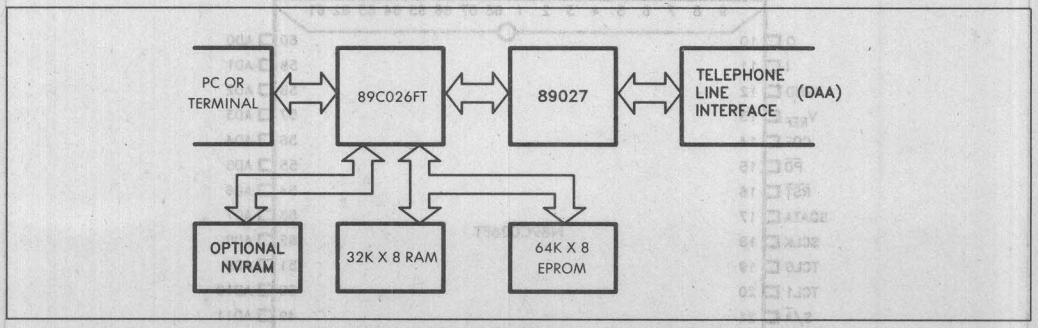


Figure 2. 89C024FT Modem Application

PACKAGING

89027 is available in PLCC and standard plastic DIP packages. The 89C026FT is available in a PLCC package. Packages are shown from top view, looking down on component side of PC board.

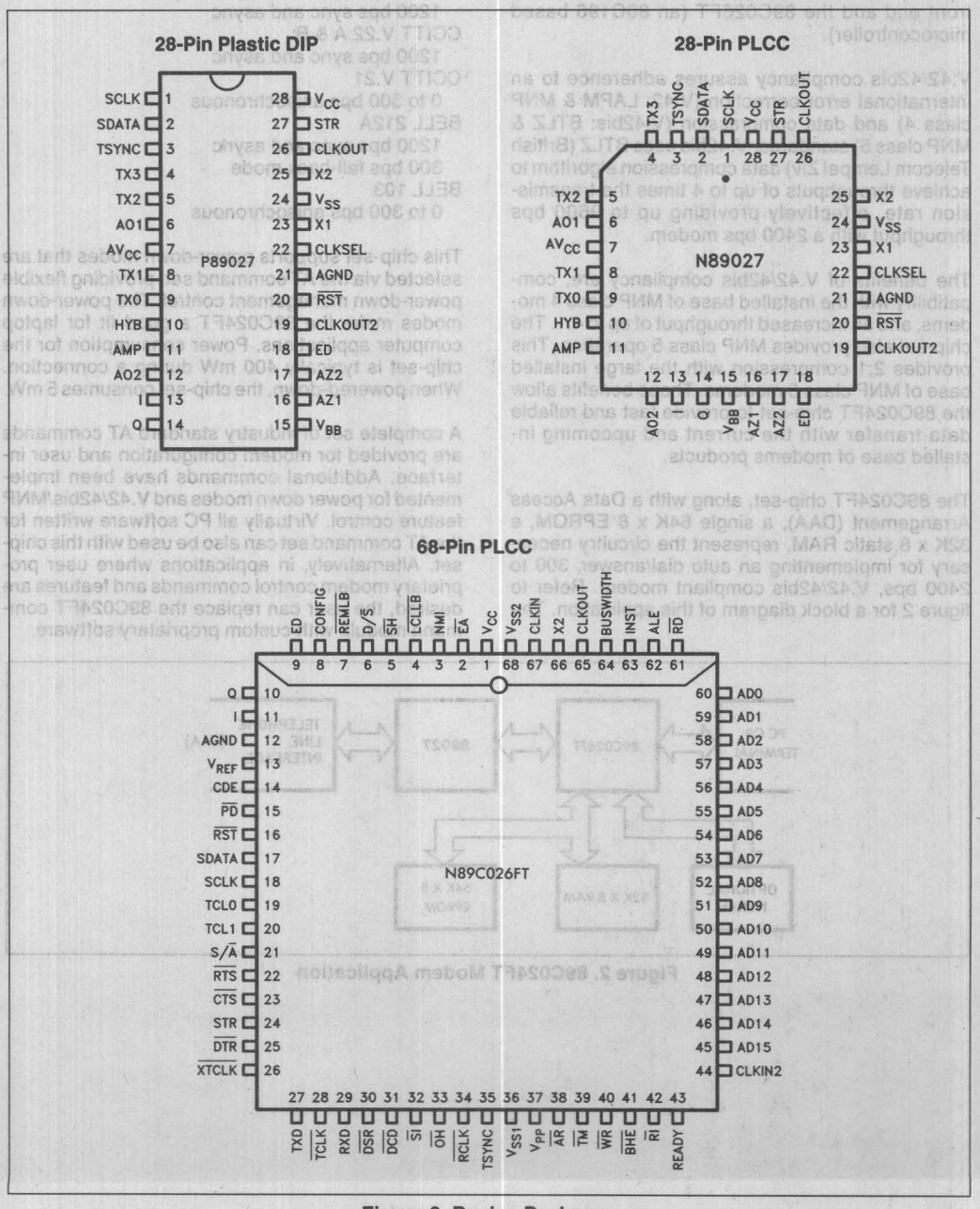


Figure 3. Device Packages

CALL ESTABLISHMENT, TERMINATION AND RETRAIN

The 89C024FT modem system incorporates all protocols and functions required for automatic or manual call establishment. The modem system also incorporates all protocols and functions required for progress and termination of a data call.

The modem chip-set has a built-in auto-dialer, both DTMF and Pulse type. It can detect dial, busy, and ringback signals from the remote end, and will provide call progress messages to the user. The modem is also capable of re-dialing the last number dialed.

The modem, when configured for auto-answer, will answer an incoming call, remain silent for the two second billing delay interval, and then transmit the answer tones. Afterwards modem to modem identification and handshaking will proceed at a speed and operating mode acceptable to both ends of the link.

The data call can also be setup by manual dialing. A transition from voice (i.e., for the purpose of manual dialing) to data mode can be done by the use of a mechanical switch (exclusion key) on the $\overline{SH}$ pin. Once set to data mode, the modem handshaking will proceed before the modems will be ready to accept and exchange data.

During data transmission, if one of the modems finds that the received data is likely to have a high bit error rate (indicated by a large mean square error in the adaptive equalizer), it initiates a retrain sequence. This automatic retrain feature is only available at 2400 bps, and is compatible with CCITT V.22 bis recommendations.

Disconnection of the data call can be initiated by the DTE at the local end or by the remote DTE (if the modem is configured to accept it). Whether $\overline{DTR}$ will initiate a disconnect depends on the last &D command. Receiving a long space from a remote modem will initiate a disconnect only after a Y1 command. The optional disconnect requests, originated by the remote modem, are of two types, (1) disconnect when receiving long-space, and (2) disconnect when received carrier is dropped. The modem chip-set can also be configured to transmit "long-space" just before disconnection.

Because the CCITT and Bell modem connection protocols do not provide recognition of remote modem type (i.e., V.22 bis to 212A), the Intel chip-set provides the additional capability of identifying the remote modem type. This feature is beneficial during the migration phase of the technology from the 1200 bps to 2400 bps. In North America, where the installed base of 1200 bps modems is mostly made-up of 212A type, this feature allows a "Data Base Service Provider" to easily upgrade the existing 212A modems to 2400 bps V.22 bis standard, transparently, to 212A users. Similarly, a user with a 89C024FT based modem system can automatically call data bases with either 212A or V.22 bis modems, without concern over the difference. This feature's benefits are realized in smooth upgrading of data links, with minimum cost and reduced disruption in services. Refer to Table 1 for a detailed description of remote modem compatibility.

SOFTWARE CONFIGURATION COMMANDS

This section lists the 89C024FT commands and registers that may be used while configuring the modem. Commands instruct the modem to perform an action, the value in the associated registers determine how the commands are performed, and the result codes returned by the modem tell the user about the execution of the commands.

The commands may be entered separately or in string fashion. Any spaces within or between commands will be ignored by the modem. During the entry of any command, the 'backspace' key (CNTRL H) can be used to correct any error. Upper case or lower case characters can be used in the commands. Commands described in the following paragraphs refer to asynchronous terminals using ASCII codes.

Table 1. Remote Modem Compatibility

Originating 89C024FT Modem		Answering Modem: Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	300	—	300*	300*
	1200	1200*	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	1200*	1200	—	1200	1200
	2400	1200*	1200		1200	2400

Answering 89C024FT Modem		Originating Modem: Bell 300	Bell 1200	CCITT 300	CCITT 1200	CCITT 2400
Bell	300	300	1200	—	1200	1200
	1200	300	1200	—	1200	1200
CCITT	300	—	—	300	—	—
	1200	300*	1200	—	1200	1200
	2400	300*	1200	—	1200	2400

* These connection data rates are obtained when connecting 89C024FT based modems end to end. The same results may not be obtained when a 89C024FT based modem is connected to other modems.

Command Set

AT	Attention code.
A	Go off-hook in answer mode
A/	Repeat previous command string
Bn	BELL/CCITT Protocol Compatibility at 300 and 1200 bps
Ds	The dialing commands (0-9 A B C D * # P R T S W , ; @)
En	Echo command (En)
Hn	Switch-Hook Control
	If &J1 option is selected, H1 will also switch the auxiliary relay
In	Request Product Code and Checksum
Ln	Speaker Volume
Mn	Monitor On/Off
Nn	Maximum Line (DCE) rate
O	On-Line
Qn	Result Codes
Sn=x	Write S Register
Sn?	Read S Register
Vn	Enable Short-Form Result Codes
Xn	Enable Extended Result Code
Yn	Enable Long Space Disconnect
Z	Fetch Configuration Profile
+++	The Default Escape Code
&Cn	DCD Options
&Dn	DTR Options
&Fn	Fetch Factory Configuration Profile
&Gn	Guard Tone
&Jn	Telephone Jack Selection
&Ln	Leased/Dial-up Line Selection
&Mn	Async/Sync Mode Selection
&Pn	Make/Break Pulse Ratio
&Rn	RTS/CTS Options
&Sn	DSR Options
&Tn	Test Commands
&Wn	Write Configuration to Non Volatile Memory
&Xn	Sync Clock Source
&Yn	Default NVRAM Profile Select
&Zn	Store Telephone Number

V.42/42bis Feature Control Commands

-Jn	V.42 Detection Phase Control
"Hn	V.42bis Compression Control
"Nn	V.42bis Dictionary Size
"On	V.42bis Dictionary String Length

MNP Feature Control Command Set

\An	Maximum MNP Block Size
%An	Set Auto-Reliable Fallback Character
\Bn	Transmit Break
\Cn	Set Auto-Reliable Buffer
\Gn	Set Modem Port Flow Control
\Jn	Bits per Second Rate Adjust
\Kn	Set Break Control
\Nn	Set Operating Mode
\O	Originate Reliable Link
\Qn	Set Serial Port Flow Control
\Tn	Set Inactivity Timer
\U	Accept Reliable Link
\Vn	Modify Result Code Form
\Xn	Set XON/XOFF Pass-Through
\Y	Switch to Reliable Mode
\Z	Switch to Normal Mode

Power Down Commands

+En	Disable/Enable Power Down
+Tn	Time to Power Down

CONFIGURATION REGISTERS

The modem stores all the configuration information in a set of registers. Some registers are dedicated to a special command and function, and others are bit-mapped, with different commands sharing the register space to store the command status.

S0*	Ring to Answer
S1	Ring Count. (Read Only)
S2	Escape Code Character
S3	Carriage Return Character
S4	Line Feed Character
S5	Back Space Character
S6	Wait for Dial Tone
S7	Wait for Data Carrier
S8	Pause Time for the Comma Dial Modifier
S9	Carrier Detect Response Time
S10	Lost Carrier to Hang Up Delay
S11*	DTMF Tone Duration
S12	Escape Code Guard Time
S13	Not Used
S14*	Bit Mapped Option Register
S15	Not Used
S16	Modem Test Options
S17	Not Used
S18*	Test Timer
S19	Not Used
S20	Not Used
S21*	Bit Mapped Options Register
S22*	Bit Mapped Options Register
S23*	Bit Mapped Options Register
S24	Not Used
S25*	Delay to DTR (Sync Only)
S26*	RTS to CTS Delay (Half Dup.)
S27*	Bit Mapped Options Register
S31*	Bit Mapped Options Register
S37	Maximum Line (DCE) Rate
S100	Mean Error Monitor Register

NOTE:
* These S registers can be stored in the NVRAM.

DIALING

Dial modifiers are available for adding conditions to dialed phone numbers.

Dial Modifiers

P	Pulse Dial
R	Originate call in Answer Mode
T	Tone Dial
S	Dial a stored number
W	Wait for dial tone
,	Delay a dial sequence
.	Return to command state
!	Initiate a flash
@	Wait for quiet

Example:

Terminal: AT &Z0 = T 1 (602) 555-1212
Modem: OK

Result: Modem stores the Tone Dial (T) modifier and phone number T16025551212 in the external NVRAM.

The number can be dialed from asynchronous mode by issuing the following command:

Terminal: AT DS0
Modem: T16025551212

Result: Modem dials phone number and attempts to establish a connection.

or by turning on $\overline{DTR}$ when in Synchronous Mode 2. Up to 33 symbols (dial digits and dial modifiers) may be stored. Spaces and other delimiters are ignored and do not need to be included in the count. If more than 33 symbols are supplied, the dial string will be truncated to 33.

POWER MANAGEMENT

The flexible power management controls allow for a variety of command and hardware driver options. The power down sequence is initiated by placing a logic "low" on pin 15 ($\overline{PD}$) of the 89C026FT. The laptop can control the $\overline{PD}$ signal directly. If such a signal is unavailable, $\overline{PD}$ can be controlled by communications software via $\overline{DTR}$. Lack of data activity or an in-coming ring signal can also be used to control $\overline{PD}$.

Placing the crystal on the 89C026FT (Figure 10) allows it to reduce power consumption by turning off the oscillator. When online and connected to a remote modem, the power consumption for the 89C024FT is typically 400 mW. Additionally, when the 89027 is not needed (on-hook, not connected to a remote modem) the 89C026FT places it in stand-by. In standby the chip-set power consumption is typically 255 mW. When powered down via the $\overline{PD}$ pin on the 89C026FT, the chip-set typically consumes 5 mW. Minimum memory-system power-consumption can be achieved by chip selecting memory only when addressed.

APPLICATIONS OVERVIEW

The block diagram of a stand-alone 300 to 2400 bps Hayes compatible modem is depicted in Figure 4. The DAA section shown in this diagram may be implemented using the suggested diagram in Figure 5. Figure 6 shows the use of the power-down feature.

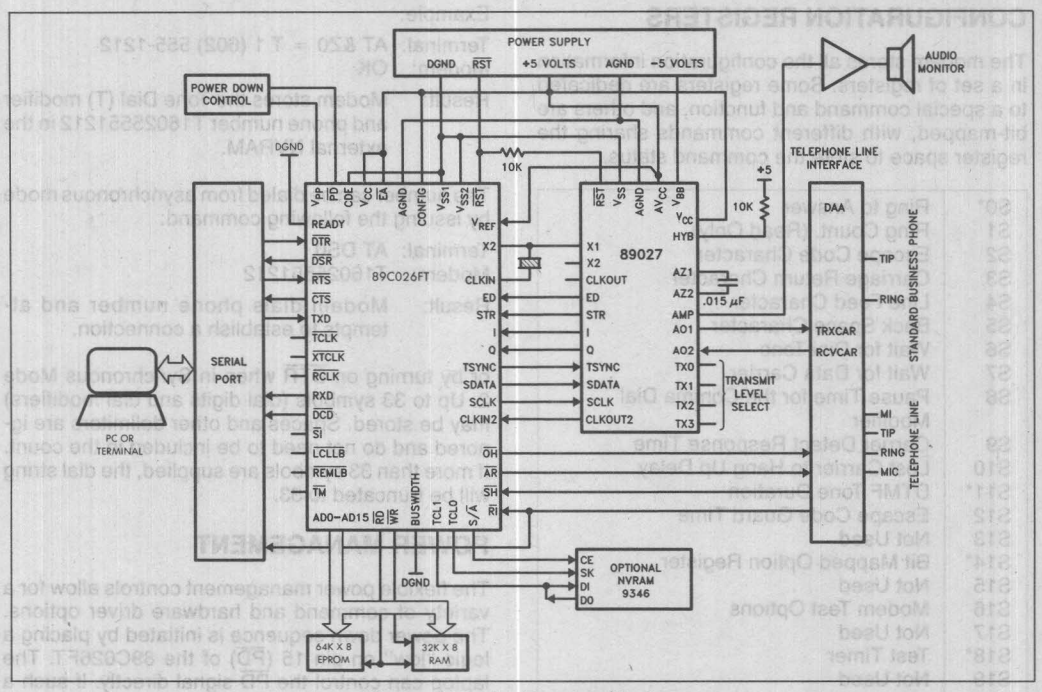


Figure 4. Typical Laptop Modem

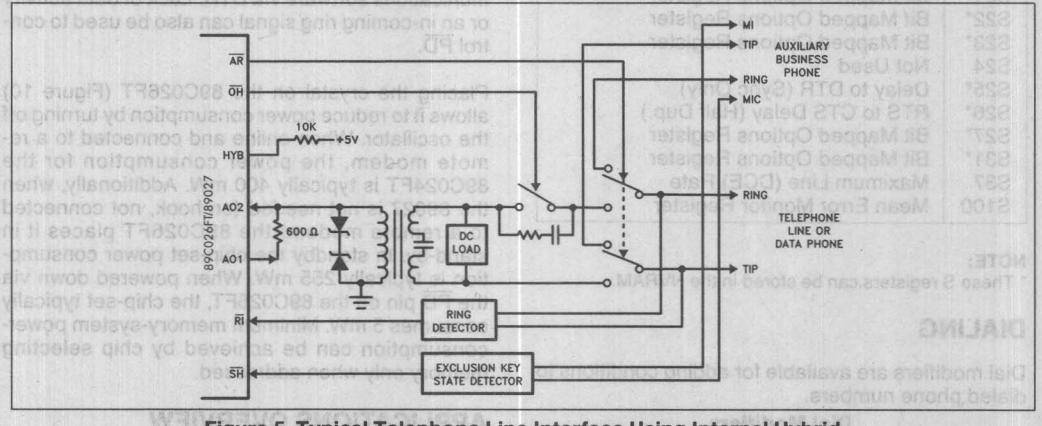


Figure 5. Typical Telephone Line Interface Using Internal Hybrid

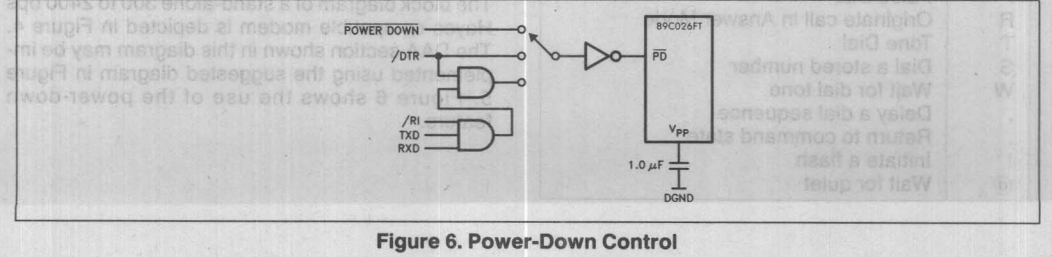


Figure 6. Power-Down Control

SYSTEM COMPATIBILITY SPECIFICATIONS

Parameter	Specification
Synchronous	2400 bps ±0.01% V.22 bis 1200 bps ±0.01% V.22 and BELL 212A
Asynchronous	2400, 1200 bps, character asynchronous. 0 - 300 bps anisochronous.
Asynchronous Speed Range	+1% −2.5% default. Extended +2.3% −2.5% range of CCITT standards optional via software customization.
Asynchronous Format	10 bits, including start, stop, parity. (8, 9, 11 bits optional via S/W customization.)
Synchronous Timing Source	a) Internal, derived from the local oscillator. b) External, provided by DTE through $\overline{\text{XTCLK}}$. c) Slave, derived from the received clock.
Telephone Line Interface	Two wire full duplex over public switched network or 4 wire leased lines. On-chip hybrid and billing delay timers.
Modulation	V.22 bis, 16 point QAM at 600 baud. V.22 and 212A, 4 point PSK at 600 baud. V.21 and 103, binary phase coherent FSK
Output Spectral Shaping	Square root of 75% raised cosine, QAM/PSK.
Transmit Carrier Frequencies V.22 bis, V.22, 212A V.21 Bell 103 mode	 Originate 1200 Hz ± .02% Answer 2400 Hz ± .02% Originate 'space' 1180 Hz ± .02% Originate 'mark' 980 Hz ± .02% Answer 'space' 1850 Hz ± .02% Answer 'mark' 1650 Hz ± .02% Originate 'space' 1070 Hz ± .02% Originate 'mark' 1270 Hz ± .02% Answer 'space' 2020 Hz ± .02% Answer 'mark' 2225 Hz ± .02%
Received Signal Frequency Tolerance V.22 bis, V.22, 212A V.21 Bell 103	 Originate 2400 Hz ± 7 Hz Answer 1200 Hz ± 7 Hz Originate 'space' 1850 Hz ± 12 Hz Originate 'mark' 1650 Hz ± 12 Hz Answer 'space' 1180 Hz ± 12 Hz Answer 'mark' 980 Hz ± 12 Hz Originate 'space' 2020 Hz ± 12 Hz Originate 'mark' 2225 Hz ± 12 Hz Answer 'space' 1070 Hz ± 12 Hz Answer 'mark' 1270 Hz ± 12 Hz
Typical Energy Detect Sensitivity	Greater than −43 dBm ED is ON. Less than −48 dBm ED is OFF. Signal in dBm measured at AO2.
Energy Detect Hysteresis	A minimum Hysteresis of 2 dB for QAM scrambled mark.
Line Equalization	Fixed compromise equalization, transmit. Adaptive equalizer for PSK/QAM, receive.
Diagnostics Available	Local analog loopback. Local digital loopback. Remote digital loopback.
Self Test Pattern Generator	Alternate 'ones' and 'zeros' and error detector, to be used along with most loopbacks. A number indicating the bit errors detected is sent to DTE.

RECEIVER PERFORMANCE SPECIFICATIONS

Test Cases		Typical SNR for 10^{-5} BER Performance	
Data Mode	Rx Level (dBm)	Answer (dB)	Originate (dB)
V.22 bis Synchronous	−30	16	16.5
	−40	16.5	18
V.22/Bell 212A Synchronous	−30	6.5	6.5
	−40	6.5	6.5
V.21 Asynchronous	−30	9	7.5
	−40	9	8
Bell 103 Asynchronous	−30	10	11.5
	−40	10	11.5

Test Conditions:

— Receive Signal (Rx) measured at A02 (transmit level set at −9 dBm)
— Unconditioned 3002 Line
— 3 KHz Flat-Band Noise

PERFORMANCE SPECIFICATIONS

Parameter	Min	Typ	Max	Units	Comments
DTMF Level		4.0		dBm	at AO1
DTMF Second Harmonic			−35	dB	HYB enabled into 600Ω
DTMF Twist (Balance)		3		dB	
Default DTMF Duration		100		ms	Software Controlled
Pulse Dialing Rate		10/20		pps	Software Controlled
Pulse Dialing Make/Break		39/61 33/67		% %	US UK, Hong Kong
Pulse Interdigit Interval		785		ms	
Billing Delay Interval			2.1	sec	
Guard Tone Frequency Amplitude Frequency Amplitude		540 −3 1800 −6		Hz dB Hz dB	referenced to High Channel transmit. QAM/PSK Modes Only
Dial Tone Detect Duration		3.0		sec	
Ringback Tone Detect Duration Cadence		 0.75 1.5		 sec 	 Off/On Ratio
Busy Tone Detect Duration Cadence	 0.67	 0.2 	 1.5	 sec 	 Off/On Ratio

89C026FT OVERVIEW

The 89C026FT processor performs data manipulation, signal processing and user interface functions. It requires a single 64K × 8 ROM and 32K × 8 RAM to execute standard, and/or custom code to perform the V.42/42bis and MNP4/5 protocol functions. The ROM and RAM addresses overlap in external memory and are decoded using the INST and AD15 signals. A block diagram of the 89C026FT is provided in Figure 7.

89C026FT contains a TTL compatible serial link to DTE equipment, along with a full complement of V.24/RS-232-C control signals. A UART or USART may be used to transfer data to and from a microcomputer bus. The 89C026FT supports the industry standard AT command set facilitating compatibility with most PC software.

During transmit operation, the 89C026FT synthesizes DTMF tones and the 300 BPS FSK modem signal and transmits them to the 89027 as digitized amplitude samples. During 1200 and 2400 BPS operation, DPSK and QAM is used to send 2 to 4 bits of information respectively at 600 baud to the AFE. Because the QAM coding technique is an inherently synchronous transmission mechanism, in the case of asynchronous QAM transmission, the asynchronous data is synchronized by adding or deleting stop bits. Following the synchronization process, the 89C026FT transmits digitized phase and amplitude samples to 89027 over the high speed serial link.

In the receive operation, the information is received by the 89C026FT from the 89027 as two signals which are 90 degrees phase shifted from each other. These analog signals are then digitized by the A/D converter resident on the 89C026FT. By using DSP algorithms, the received signals are processed using adaptive equalization for telephone line delay, amplitude distortion and gain adjustments and the signal demodulated. Following demodulation, the data is unscrambled, and if necessary, returned to asynchronous format.

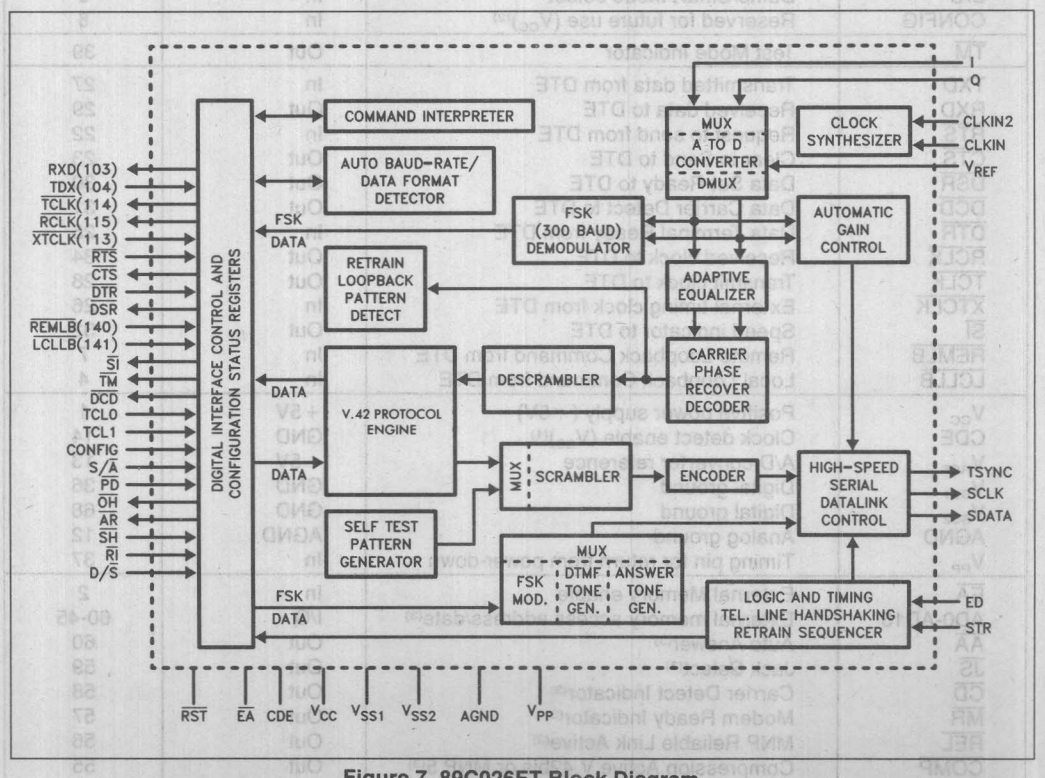


Figure 7. 89C026FT Block Diagram

89C026FT PINOUT

Symbol	Function (89C026FT)	Direction[4]	Pin No.
CLKIN	12.96 MHz master clock from 89027	In	67
CLKIN2	270 KHz from 89027	In	44
$\overline{RST}$	Chip reset (active low)	In	16
I	In-phase received signal	In	11
Q	Quadrature-phase received signal	In	10
STR	Symbol Timing from 89027	In	24
ED	Energy Defect input	In	9
TSYNC	Transmitter sync pulse to 89027	Out	35
SDATA	Serial Data to 89027	Out	17
SCLK	Serial Clock to 89027	Out	18
$\overline{OH}$	Off-Hook control to DAA	Out	33
$\overline{SH}$	Switch-Hook from dataphone	In	5
$\overline{RI}$	Ring Indicator from DAA	In	42
$\overline{AR}$	Aux Relay control to DAA	Out	38
TCL1	NVRAM Data I/O	I/O	20
TCL0	NVRAM CLK	Out	19
$\overline{PD}$	Power-down control	In	15
S/$\overline{A}$	NVRAM CE	Out	21
D/$\overline{S}$	Dumb/$\overline{Smart}$ mode select	In	6
CONFIG	Reserved for future use (V_{CC})[2]	In	8
$\overline{TM}$	Test Mode Indicator	Out	39
TXD	Transmitted data from DTE	In	27
RXD	Received data to DTE	Out	29
$\overline{RTS}$	Request to send from DTE	In	22
$\overline{CTS}$	Clear to Send to DTE	Out	23
$\overline{DSR}$	Data Set Ready to DTE	Out	30
$\overline{DCD}$	Data Carrier Detect to DTE	Out	31
$\overline{DTR}$	Data Terminal Ready from DTE	In	25
$\overline{RCLK}$	Received clock to DTE	Out	34
$\overline{TCLK}$	Transmit clock to DTE	Out	28
$\overline{XTCLK}$	External timing clock from DTE	In	26
$\overline{SI}$	Speed indicator to DTE	Out	32
$\overline{REMLB}$	Remote Loopback Command from DTE	In	7
$\overline{LCLLB}$	Local Loopback Command from DTE	In	4
V_{CC}	Positive power supply (+5V)	+5V	1
CDE	Clock detect enable (V_{SS})[1]	GND	14
V_{REF}	A/D converter reference	+5V	13
V_{SS1}	Digital ground	GND	36
V_{SS2}	Digital ground	GND	68
AGND	Analog ground	AGND	12
V_{PP}	Timing pin for return from power-down	In	37
$\overline{EA}$	External Memory enable	In	2
AD0-AD15	External memory access address/date[3]	I/O	60-45
$\overline{AA}$	Auto Answer[3]	Out	60
$\overline{JS}$	Jack Select[3]	Out	59
$\overline{CD}$	Carrier Detect Indicator[3]	Out	58
$\overline{MR}$	Modem Ready Indicator[3]	Out	57
$\overline{REL}$	MNP Reliable Link Active[3]	Out	56
$\overline{COMP}$	Compression Active V.42bis or MNP 5[3]	Out	55
$\overline{ERR}$	Error detected by LAPM or MNP[3]	Out	54
$\overline{LAPM}$	LAPM Reliable Link Active[3]	Out	53

89C026FT PINOUT (Continued)

Symbol	Function (89C026FT)	Direction(4)	Pin No.
NMI	Non-maskable interrupt(V_{SS})(1)	In	3
X2	Crystal output	Out	66
CLKOUT	Clk output	Out	65
BUSWIDTH	Bus Width	In	64
INST	External memory instruction fetch	Out	63
ALE	Address latch enable	Out	62
$\overline{RD}$	External memory read	Out	61
READY	External memory ready	In	43
$\overline{BHE}$	External memory bus high enable	Out	41
$\overline{WR}$	External memory write	Out	40

NOTES:
1. Pins marked with (V_{SS}) must be corrected to V_{SS}.
2. Pins marked with (V_{CC}) must be connected to V_{CC}.
3. AD0-AD3 are used as $\overline{AA}$, $\overline{JS}$, $\overline{CD}$, $\overline{MR}$, $\overline{REL}$, $\overline{COMP}$, $\overline{ERR}$, and $\overline{LAPM}$ respectively.
4. Pins with direction "IN" must not be left floating.

89C026FT PIN DESCRIPTION

$\overline{XTCLK}$

Transmitter timing from DTE, when external clock option is selected.

TXD

The serial data from DTE to be transmitted on the line. A logic 'high' is mark. In synchronous mode, 89C026FT samples this data on the rising edges of $\overline{TCLK}$.

$\overline{TCLK}$

Clock output from 89C026FT as timing source for data exchange from DTE to modem. Serial data is read on the rising edges of the $\overline{TCLK}$. This output is High in asynchronous mode.

RXD

The serial data to DTE. A logic 'high' is mark. In synchronous mode, the rising edge of $\overline{RCLK}$ occurs in the middle of RXD.

$\overline{RCLK}$

Synchronous clock output. Rising edge of $\overline{RCLK}$ occurs in the middle of each RXD bit. This pin remains High in asynchronous mode.

$\overline{PD}$

Power-down control. A low on this input pin, in conjunction with the +En and +Tn commands, will cause the modem to go into a power-down mode.

V_{PP}

Timing pin for return from power-down. Connect a *1.0 μf capacitor between V_{PP} and V_{SS} if the power-down* option is used. This capacitor causes an internal timing circuit to give the oscillator time to stabilize before turning on internal clocks. This pin may be left floating or connected through a 1.0 μF capacitor to V_{SS} if power-down mode is not required.

$\overline{TM}$

A low indicates maintenance condition in the modem.

$\overline{DCD}$

In async operation, $\overline{DCD}$ remains low regardless of data carrier (default), or it can be programmed to indicate received carrier signal is within the required timing and amplitude limits. In sync operation low indicates the received carrier signal is within the required timing and amplitude limits.

$\overline{DSR}$

A low indicates modem is off-hook, is in data transmission mode, and the answer tone is being exchanged. $\overline{CTS}$ low indicates modem is prepared to accept data.

$\overline{RTS}$

In async mode $\overline{RTS}$ is ignored. Under command control, in sync mode $\overline{RTS}$ can be ignored, or the modem can respond with a Low on $\overline{CTS}$.

$\overline{DTR}$

&D0 command will cause the modem to ignore $\overline{DTR}$. For &D1 the modem assumes the asynchronous command state on a low-to-high transition of the $\overline{DTR}$ circuit. The &D2 command does the same as &D1 except the state of $\overline{DTR}$ will enable/disable auto answer. A low-to-high transition of $\overline{DTR}$ after the &D3 command will cause the modem to assume the initialization state.

TCL1, TCL0

These pins are used as the serial clock and data for interface to an NVRAM. Refer to Figure 3. TCL0 is used to output a clock and serial data is transferred in on TCL1.

$\overline{AR}$

This Auxiliary relay control is for switching a relay for voice or data calls. High is voice, low is data.

$\overline{RI}$

A low signal from DAA indicates line ringing. This input is ignored when the modem is configured for leased line. This signal should follow the ring cadence.

$\overline{OH}$

Low sets an off hook condition, high indicates an on hook. When dialing, this signal is used to pulse dial the line.

$\overline{SH}$

Used as a telephone voice to data switch or vice versa. Any logic level transition will toggle the modem state between voice and data.

$\overline{AA}$

Used as an indicator for Auto Answer status and Ring indicator. Active low.

$\overline{LCLLB}$

A low will set the modem in the local analog loopback test mode. Logic Low levels applied simultaneously to $\overline{REMLB}$ and $\overline{LCLLB}$ pins, sets the modem to the local digital loopback.

$\overline{REMLB}$

A low on this pin initiates a remote loopback condition.

$\overline{CD}$

A low indicates the presence of carrier signal on the line.

$\overline{MR}$

A low indicates the presence of the DSR signal. Toggling indicates that a test mode is active.

$\overline{REL}$

A low indicates that an MNP reliable link has been established.

$\overline{COMP}$

A low indicates that data compression is in operation (V.42bis or MNP Class 5).

$\overline{LAPM}$

A low indicates that a LAPM reliable link has been established.

$\overline{ERR}$

Goes low for 1 second whenever a reliable connection detects an error.

$\overline{SI}$

Selects one of the two data rates or ranges of rates in the DTE to correspond to the rate in modem. Low selects the higher rate (2400 CCITT/1200 BELL) or range of rates. High selects the Low rate or range of rates.

D/$\overline{S}$

A low on this pin will indicate the smart mode which will respond to all commands. A High will ignore all commands.

V_{REF}

Voltage reference for the analog to digital converter should be connected to the 89027 AV_{CC}.

CDE

This pin must be connected to V_{SS}.

S/$\overline{A}$

The function of this pin is re-defined as external NVRAM CE.

CONFIG

Reserved for future use. This signal should be pulled high.

$\overline{EA}$

When high, memory access from address 2000H to 4000H are directed to on-chip ROM. When low, all Memory access is directed to off-chip memory. This pin must be tied high.

$\overline{JS}$

Low is used to pulse A and A1 leads to control a 1A2 Key System jack.

BUSWIDTH

When high, external memory accesses are 16 bits wide. When low, external memory accesses are 8 bits wide. This pin must be tied low.

READY

When high, no wait states are inserted in external memory accesses. When low, one wait state is inserted in each external memory access.

INST

Output high during an external memory read indicates the read is an instruction fetch. INST is activated only during external memory accesses and output low for data fetch. INST along with AD15 are used to decode the overlapping external ROM and RAM.

89C026FT ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... 0°C to +70°C
Storage Temperature ..... −40°C to +125°C
Voltage from Any Pin to V_{SS} or AGND ..... −0.5V to +7.0V
Average Output Current from Any Pin ..... 10 mA
Power Dissipation ..... 1.5 Watts

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

NOTICE: Specifications contained within the following tables are subject to change.

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T_A	Ambient Temperature Under Bias	0	+70	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V
V_{REF}	Analog Supply Voltage	4.75	5.25	V
f_{OSC}	CLKIN Frequency	12.95870	12.96130	MHz

NOTE:
The AGND and V_{SS} on both the 89C026FT and the 89027 must be nominally at the same potential.

4

D.C. CHARACTERISTICS

Symbol	Parameter	Min	Typ(7)	Max	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5		+0.8	V	
V_{IH}	Input High Voltage(1)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage on CLKIN	0.7 V_{CC}		V_{CC} + 0.5	V	
V_{IH2}	Input High Voltage on RESET	2.2		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage			0.3 0.45 1.5	V V V	I_{OL} = 200 μA I_{OL} = 3.2 mA I_{OL} = 7 mA
V_{OH}	Output High Voltage(4)	V_{CC} − 0.3 V_{CC} − 0.7 V_{CC} − 1.5			V V V	I_{OH} = −200 μA I_{OH} = −3.2 mA I_{OH} = −7 mA
V_{OH1}	Output High Voltage(3)	V_{CC} − 0.3 V_{CC} − 0.7 V_{CC} − 1.5			V V V	I_{OH} = −10 μA I_{OH} = −30 μA I_{OH} = −60 μA
I_{LI}	Input Leakage Current(5)			±10	μA	0 < V_{IN} < V_{CC} − 0.3V
I_{LI1}	Input Leakage Current(6)			±3	μA	0 < V_{IN} < V_{REF}
I_{IL}	Logical 0 Input Current(3)			−50	μA	V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current in RESET(2) (ALE, $\overline{RD}$, $\overline{WR}$, $\overline{BHE}$, INST, SCLK)			−850	μA	V_{IN} = 0.45V

D.C. CHARACTERISTICS (Continued)

Symbol	Parameter	Min	Typ(7)	Max	Units	Test Conditions
I_{REF}	A/D Converter Reference Current		2	5	mA	CLKIN = 12.96 MHz $V_{CC} = V_{PP} = V_{REF} = 5.25$
I_{CC1}	Active Mode Current (Typical)		45	60	mA	CLKIN = 12.96 MHz
R_{RST}	RESET Pullup Resistor	6K		50K	Ω	
C_S	Pin Capacitance (Any Pin to V_{SS})			10	pF	f_{TEST} = 1.0 MHz
I_{PD}	Power-Down Mode Current		5	50	μA	$V_{CC} = V_{PP} = V_{REF} = 5.25$

NOTES:
(Notes apply to all specifications)
1. All pins except RESET and CLKIN.
2. Holding these pins below V_{IH} in RESET may cause the part to enter test modes.
3. TCL0, TCL1, S/A, RTS, CTS, DSR, DCD, SI, OH.
4. BHE, INST, CLKOUT, RESET, TCLK, RXD, RCLK, TSYNC, TM, SCLK, SDATA. The V_{OH} specification is not valid for RESET.
5. CDE, EA, READY, BUSWIDTH, NMI, STR, DTR, XTCLK, TXD, B/C, CLKIN2, and RI.
6. S/D, SH, REMLB, LCLLB, I, Q, CONFIG, ED.
7. Typicals are based on a limited number of samples and are not guaranteed. The values listed are at room temperature and $V_{REF} = V_{CC} = 5V$.

A.C. CHARACTERISTICS (Over specified operating conditions)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall times = 10 ns, f_{OSC} 12.96 MHz

This system must meet these specifications to work with 89C026FT:

Symbol	Parameter	Min	Max	Units	Notes
T_{AVYV}	Address Valid to READY Setup		$2T_{OSC} - 85$	ns	
T_{LLYV}	ALE Low to READY Setup		$T_{OSC} - 72$	ns	
T_{YLYH}	Non READY Time	No Upper Limit		ns	
T_{CLYX}	READY Hold after CLKOUT Low	0	$T_{OSC} - 30$	ns	(Note 1)
T_{LLYX}	READY Hold after ALE Low	$T_{OSC} - 15$	$2T_{OSC} - 40$	ns	(Note 1)
T_{AVGV}	Address Valid to Buswidth Setup		$2T_{OSC} - 85$	ns	
T_{LLGV}	ALE Low to Buswidth Setup		$T_{OSC} - 70$	ns	
T_{CLGX}	Buswidth Hold after CLKOUT Low	0		ns	
T_{AVDV}	Address Valid to Input Data Valid		$3T_{OSC} - 67$	ns	
T_{RLDV}	RD Active to Input Data Valid		$T_{OSC} - 23$	ns	
T_{CLDV}	CLKOUT Low to Input Data Valid		$T_{OSC} - 50$	ns	
T_{RHDZ}	End of RD to Input Data Float		$T_{OSC} - 20$	ns	
T_{RXDX}	Data Hold after RD Inactive	0		ns	

NOTE:
1. If max is exceeded, additional wait states will occur.

A.C. CHARACTERISTICS (Continued)

Test Conditions: Capacitive load on all pins = 100 pF, Rise and fall tines = 10 ns, f_{OSC} 12.96 MHz

The 89C026FT will meet these specifications:

Symbol	Parameter	Min	Max	Units	Notes
F_{CLKIN}	Oscillator Frequency	12.95870	12.96130	MHz	
T_{OSC}	Oscillator Period	$1/F_{CLKIN(MAX)}$	$1/F_{CLKIN(MIN)}$	ns	
T_{XHCH}	F_{CLKIN} High to CLKOUT High or Low	40	110	ns	(Note 1)
T_{CLCL}	CLKOUT Cycle Time	$2\ T_{OSC}$		ns	
T_{CHCL}	CLKOUT High Period	$T_{OSC} - 10$	$T_{OSC} + 10$	ns	
T_{CLLH}	CLKOUT Falling Edge to ALE Rising	−5	15	ns	
T_{LLCH}	ALE Falling Edge to CLKOUT Rising	−15	15	ns	
T_{LHLH}	ALE Cycle Time	$4\ T_{OSC}$		ns	
T_{LHLL}	ALE High Period	$T_{OSC} - 10$	$T_{OSC} + 10$	ns	
T_{AVLL}	Address Setup to ALE Falling Edge	$T_{OSC} - 20$		ns	
T_{LLAX}	Address Hold after ALE Falling Edge	$T_{OSC} - 40$		ns	
T_{LLRL}	ALE Falling Edge to $\overline{RD}$ Falling Edge	$T_{OSC} - 40$		ns	
T_{RLCL}	$\overline{RD}$ Low to CLKOUT Falling Edge	5	30	ns	
T_{RLRH}	$\overline{RD}$ Low Period	$T_{OSC} - 5$	$T_{OSC} + 25$	ns	
T_{RHLH}	$\overline{RD}$ Rising Edge to ALE Rising Edge	T_{OSC}	$T_{OSC} + 25$	ns	(Note 2)
T_{RLAZ}	$\overline{RD}$ Low to Address Float		10	ns	
T_{LLWL}	ALE Falling Edge to $\overline{WR}$ Falling Edge	$T_{OSC} - 10$		ns	
T_{CLWL}	CLKOUT Low to $\overline{WR}$ Falling Edge	0	25	ns	
T_{QVWH}	Data Stable to $\overline{WR}$ Rising Edge	$T_{OSC} - 23$		ns	
T_{CHWH}	CLKOUT High to $\overline{WR}$ Rising Edge	−10	10	ns	
T_{WLWH}	$\overline{WR}$ Low Period	$T_{OSC} - 30$	$T_{OSC} + 5$	ns	
T_{WHQX}	Data Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{WHLH}	$\overline{WR}$ Rising Edge to ALE Rising Edge	$T_{OSC} - 10$	$T_{OSC} + 15$	ns	(Note 2)
T_{WHBX}	$\overline{BHE}$, INST, Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{RHBX}	$\overline{BHE}$, INST, Hold after $\overline{RD}$ Rising Edge	$T_{OSC} - 10$		ns	
T_{WHAX}	AD8–15 Hold after $\overline{WR}$ Rising Edge	$T_{OSC} - 50$		ns	
T_{RHAX}	AD8–15 Hold after $\overline{RD}$ Rising Edge	$T_{OSC} - 25$		ns	

NOTES:
1. Typical specifications, not guaranteed.
2. Assuming back-to-back bus cycles.

WAVEFORMS

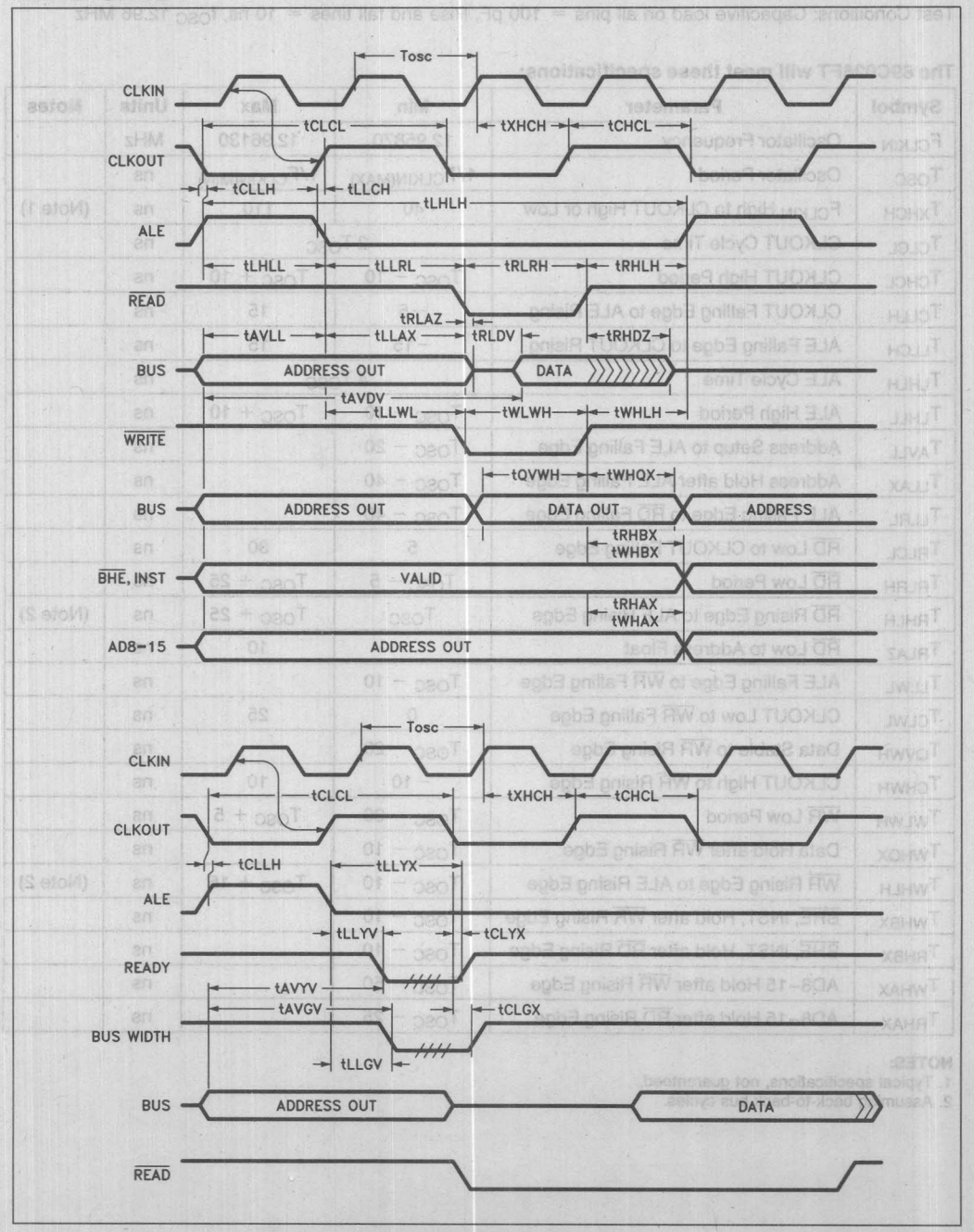


Figure 8. Bus Signal Timings

89027 OVERVIEW

The 89027 is a 28 pin CHMOS analog front end device, which performs most of the complex filtering functions required in modem transmitters and receivers. A general block diagram of this chip is provided in Figure 9. Most of the analog signal processing functions in this chip are implemented with CHMOS switched capacitor technology. The 89027 functions are controlled by 89C026FT, through a high speed serial data link.

During FSK transmit operation, the 89027 receives digitally synthesized mark and space sinusoid amplitude information from the 89C026FT. The 89027 converts the signal to its analog equivalents, filters it, and transmits it to the telephone line. For QAM transmission, the signal constellation points are transferred to the 89027. This information is modulated into an analog signal, passed through spectral shaping filters, combined with the necessary guard tone, smoothed by a low pass filter, and transmitted to the line. Prior to transmitting either FSK or QAM signals to the telephone line, the 89027 adjusts the signal gain through an on-board programmable gain amplifier.

During the receive operation, the received FSK and QAM signals are passed through anti-alias filters, bandsplit filters, automatic gain control and carrier detect circuits, a Hilbert transform filter, and the output sent to the 89C026FT processor as analog signals.

Other functions provided by the 89027 are: an on-board two wire to four wire circuit with disable capability, an audio monitor output with software configurable gain, and a programmable gain transmit signal.

The 89027 is available in 28 pin plastic DIP and PLCC packages.

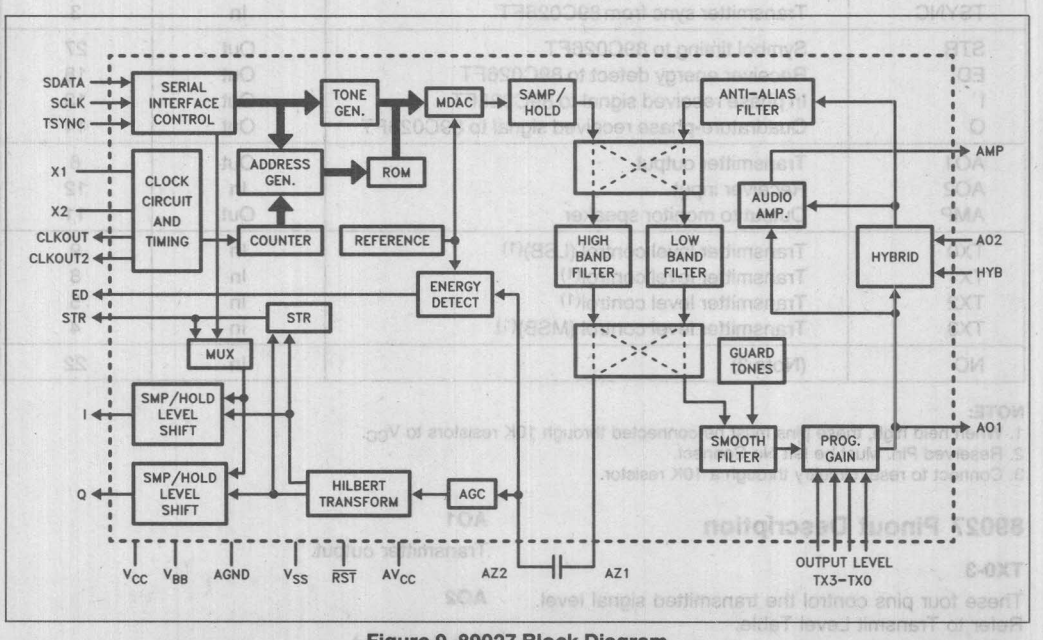


Figure 9. 89027 Block Diagram

89027 PINOUT

Symbol	Function (89027)	Direction	Pin No.
V_{CC}	Positive Power Supply (Digital)	+5V	28
V_{BB}	Negative Power Supply	−5V	15
V_{SS}	Digital Ground	DGND	24
AGND	Analog Ground	AGND	21
AV_{CC}	Positive Power Supply (Analog)	+5	7
X1	Xtal Oscillator	In	23
X2	Xtal Oscillator	Out	25
CLKOUT	12.96 MHz Clock Output to 89C026FT	Out	26
CLKOUT2	270 KHz Clock Output to 89C026FT	Out	19
$\overline{RST}$	Chip reset (active low)(3)	In	20
HYB	Enable on-chip hybrid(1)	In	10
AZ1	Auto-zero capacitor	Out	16
AZ2	Auto-zero capacitor	In	17
SDATA	Serial data from 89C026FT	In	2
SCLK	Serial clock from 89C026FT	In	1
TSYNC	Transmitter sync from 89C026FT	In	3
STR	Symbol timing to 89C026FT	Out	27
ED	Receiver energy detect to 89C026FT	Out	18
I	In phase received signal to 89C026FT	Out	13
Q	Quadrature-phase received signal to 89C026FT	Out	14
AO1	Transmitter output	Out	6
AO2	Receiver input	In	12
AMP	Output to monitor speaker	Out	11
TX0	Transmitter level control (LSB)(1)	In	9
TX1	Transmitter level control(1)	In	8
TX2	Transmitter level control(1)	In	5
TX3	Transmitter level control (MSB)(1)	In	4
NC	(Note 2)	In	22

NOTE:
1. When held high, these pins must be connected through 10K resistors to V_{CC}.
2. Reserved Pin. Must be left No Connect.
3. Connect to reset circuitry through a 10K resistor.

89027 Pinout Description

TX0-3

These four pins control the transmitted signal level. Refer to Transmit Level Table.

HYB

This pin enables the on-chip hybrid. A line impedance matching network must be connected between AO1 and AO2 when HYB is enabled. If HYB is disabled and an external 4W/2W hybrid is used, the hybrid receive path must be amplified by 6 dB.

AO1

Transmitter output.

AO2

Receiver input.

AMP

This output can be used to monitor the call progress tones and operation of the line.

ABSOLUTE MAXIMUM RATINGS(2)

Temperature Under Bias .....0 to +70° C
Storage Temperature .....−40 to +125° C
All Input and Output Voltages with Respect to V_{BB} .....−0.3V to +13.0V
All Input and Output Voltages with Respect to V_{CC} & AV_{CC} .....−13.0V to 0.3V
Power Dissipation 1.35W
Voltage with Respect to V_{SS}(1) .....−0.3V to 6.5V

NOTES:
1. Applies to pins SCLK, SDATA, TSYNC, $\overline{RST}$, HYB, TX0–TX3 only.
2. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
T_A	Ambient Temperature Under Bias	0	+70	°C
V_{CC}	Digital Supply Voltage	4.75	5.25	V
V_{BB}	Analog Supply Voltage	−4.75	−5.25	V

POWER DISSIPATION

Ambient Temp = 0° to 70° C, V_{CC} = AV_{CC} = 5 ± 5%, V_{SS} = AGND = 0V. Typical values shown are for T_A = 25° C and nominal power supplies.

Symbol	Parameter	Min	Typ	Max	Units
$AIcc_1$	AV_{CC} Operating Current		15	21	mA
Icc_1	V_{CC} Operating Current		5	6	mA
Ibb_1	V_{BB} Operating Current		−15	−21	mA
$AIcc_s$	AV_{CC} Standby Current		0.2	1	mA
Icc_s	V_{CC} Standby Current		5	6	mA
Ibb_s	V_{BB} Standby Current		−0.6	−2	mA
$AIcc_p$	AV_{CC} Power-Down Current		100		µA
Icc_p	V_{CC} Power-Down Current		450		µA
Ibb_p	V_{BB} Power-Down Current		450		µA
Pdo	Operating Power Dissipation		175	250	mW
Pds	Standby Power Dissipation		30	50	mW
Pd_p	Power Down Power Dissipation		5		mW

4

D.C. CHARACTERISTICS (T_A = 0°C to 70°C, $AV_{CC} = V_{CC}$ = 5V ± 5%, V_{BB} = 5V ± 5%, AGND = V_{SS} = 0V).

Inputs: TX0, TX1, TX2, TX3, HYB, $\overline{RST}$
Outputs: CLKOUT

Symbol	Parameter	Min	Max	Units	Test Conditions
Iil	Input Leakage Current	−10	+10	μA	$V_{SS} \leq Vin \leq V_{CC}$
Vil	Input Low Voltage	V_{SS}	0.8	V	
Vih	Input High Voltage	2.0	V_{CC}	V	
Vol	Output Low Voltage		0.4	V	Iol ≥ −1.6mA,1 TTL load
Voh	Output High Voltage	2.4		V	Ioh ≤ 50μa, 1 TTL load
Vcol	CLKOUT Low Voltage		0.4	V	Load Capacitance = 60 pF
Vcoh	CLKOUT High Voltage	0.7 V_{CC}		V	Load Capacitance = 60 pF

A.C. CHARACTERISTICS (T_A = 25°C, $V_{CC} = AV_{CC}$ = 5V, V_{SS} = AGND = 0V, V_{BB} = −5V)

ANALOG INPUTS: AO2

Parameter	Min	Typ	Max	Units	Test Condition
AO2 Receive Signal Level			−9	dBm	Hybrid Enabled
AO2 Input Resistance		10		MOhms	−2.5V<Vin<+2.5V
AO2 Allowed DC offset	−30		+30	mV	Relative to AGND

AUTO ZERO CAPACITANCE

Capacitance = 0.015 μF
Tolerance = ±20%
Voltage Rating = 10V
Type = Non-Electrolytic, low leakage.

CRYSTAL REQUIREMENTS(1)

Parameter	Min	Typ	Max	Units	Comments
Frequency Accuracy (0°C–70°C)	−0.0035%	12.96	+0.0035%	MHz	Refer to Figure 10
Rx		10	16	Ohms	
Cx		0.024		pF	
Co	5.1	5.6	6.1	pF	
C_L(2)	−5%	47	+5%	pF	2 Load Capacitors

NOTES:
1. Crystal Type: Parallel Resonant
2. Crystal manufacturers usually specify the accuracy of a parallel resonant circuit at a given "load capacitance". This "load capacitance" is specified to the crystal manufacturer as 33 pf. 33 pf includes the parallel combination of the capacitances seen at the pins of the crystal. These capacitances include C_L, IC pin capacitances, and a 3 ± 2 pf trace capacitance.

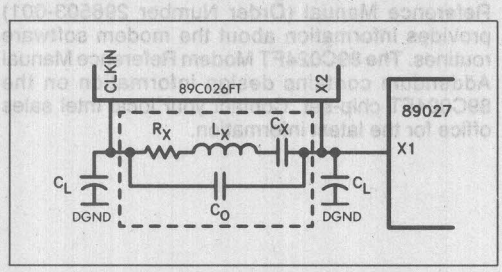


Figure 10. Crystal Equivalent Circuit

ANALOG OUTPUTS: A01, AMP

Parameter	Min	Typ	Max	Units	Comments
Load Resistance AO1 AMP	 600 10			 Ohms kOhms	
Load Capacitance AMP			 100	 pF	
Audio Amp Gain AO1 to Amp		−9 −18 −26 −70		dB dB dB dB	Max Mid Min Off (Software Selectable)
Audio Amp Gain(1) AO2 to Amp		+12 +3 −4 −60		dB dB dB dB	Max Mid Min Off (Software Selectable)

NOTE:
1. Assumes on-chip hybrid is enabled. When on-chip hybrid is disabled, gain with respect to AO2 is reduced by 6 dB.

Transmit Output Level(1)		
TX 3,2,1,0	Typ	Units
0 0 0 0	+5	dBm
0 0 0 1	+4	dBm
•	•	•
•	•	•
•	•	•
1 1 1 0	−9	dBm
1 1 1 1	−10	dBm

NOTE:
1. For PSK and QAM transmit signal. For FSK transmit signal levels, they are typically 1 dB lower. All signals are measured at A01. The tolerance for the transmit levels is ±1 dBm.

89C024LT CHIP-SET USERS:

An example of how to design a single board for both an 89C024LT and 89C024FT modem product is available from your local Intel sales office. This document is titled "Upgrading From the 89C024LT to 89C024FT." It provides information for a smooth upgrade from an 89C024LT to an 89C024FT product.

REFERENCE MANUALS

The Modem Reference Manual (Order Number 296235) contains pin descriptions, AT and MNP command descriptions, schematics, and important design guidelines for the 89C024LT, 89C024XE, and 89024 modem chip-sets. The Modem Software Reference Manual (Order Number 296503-001) provides information about the modem software routines. The 89C024FT Modem Reference Manual Addendum contains design information on the 89C024FT chip-set. Contact your local Intel sales office for the latest information.

MD024LT
MODEM DEMO BOARD

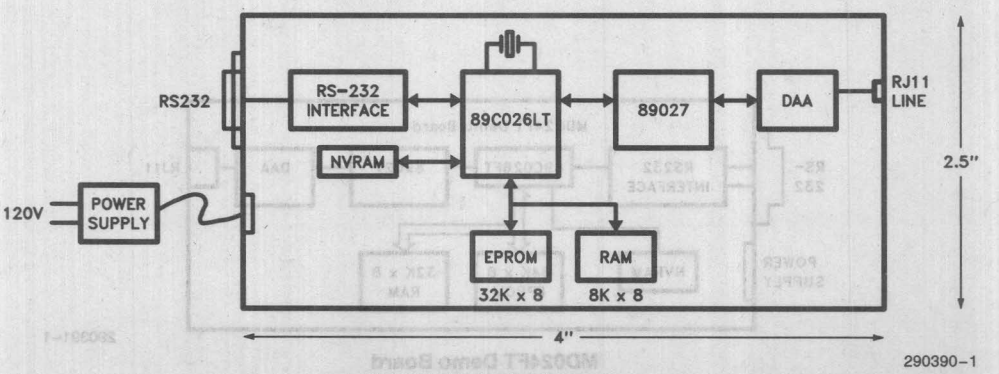
- **Fully Functional Modem**
- **Demonstrates 89C024LT Operation**
- **Pocket Modem Size (4" by 2.5")**
- **MNP* Class 4 Error Correction**
- **MNP Class 5 Data Compression**
- **RS232 Serial Interface**
- **Includes Power Supply**
- **Includes Demonstration Software**

The MD024LT is a modem demo board in a pocket modem style. It facilitates simple demonstration of the 89C024LT operation. The 89C024LT is a low power, error correcting Laptop modem chip-set. The 89C024LT is composed of the 89027 Analog Front End and the 89C026LT Microcontroller.

The MD024LT includes a 32k x 8 EPROM and an 8k x 8 SRAM to demonstrate the AT auto-dialer, MNP class 4 error correction and MNP class 5 data compression. Phone numbers and parameters may be stored in the on-board NVRAM. A terminal can be used to configure the modem via the RS232 serial port. A DAA is provided for direct phone hook up via an RJ11 connector.

The MD024LT board is shipped with a 120V power supply, demonstration utility diskette for an IBM Personal computer, a demonstration kit document, and an RJ11 phone cord. The diskette contains a program to demonstrate and measure the 89C024LT throughput, and a copy of Intel's communication software iTerm.

4



MD024LT Block Diagram

*MNP is a registered trademark of Microcom Inc.

September 1990
Order Number: 290390-001

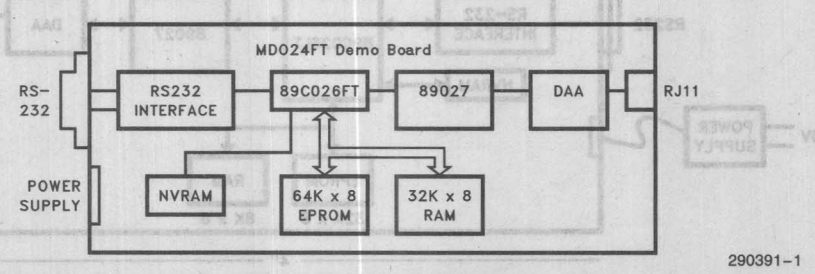
MD024FT
MODEM DEMO BOARD

- **Fully Functional Modem**
- **Demonstrates 89C024FT Operation**
- **Pocket Modem Size (4.5″ by 2.5″)**
- **V.42 Compliant Error Correction Protocols (LAPM and MNP*4)**
- **V.42bis and MNP 5 Data Compression Increase throughput up to 4 Times with DTE Rates of 9600**
- **RS232 Serial Interface**
- **Includes Power Supply**
- **Includes Demonstration Software**

The MD024FT is a modem demo board in a pocket modem style. It facilitates simple demonstration of the 89C024FT operation. The 89C024FT is a low power V.42/42bis compliant Laptop modem chip-set. The 89C024FT is composed of the 89027 Analog Front End and the 89C026FT Microcontroller.

The MD024FT includes a 64k by 8 EPROM and a 32k by 8 SRAM to demonstrate the AT auto-dialer, V.42 error correction (LAPM and MNP class 4) and V.42bis data compression (BTLZ and MNP class 5). Phone numbers and parameters may be stored in the optional on-board NVRAM. A terminal or PC can be used to configure the modem via the RS232 serial port. A DAA is also provided for direct phone hook up via an RJ11 connector.

The MD024FT board is shipped with a 120V AC power supply, demonstration utility diskette for an IBM Personal computer and an RJ11 phone cord. The utility diskette contains a program to demonstrate and measure the 89C024FT's throughput, and a copy of Intel's communication software, iTerm.



MD024FT Demo Board

*MNP is a registered trademark of Microcom Inc.

September 1990
Order Number: 290391-001

89024 MEK2
89024 ENHANCED MODEM EVALUATION KIT

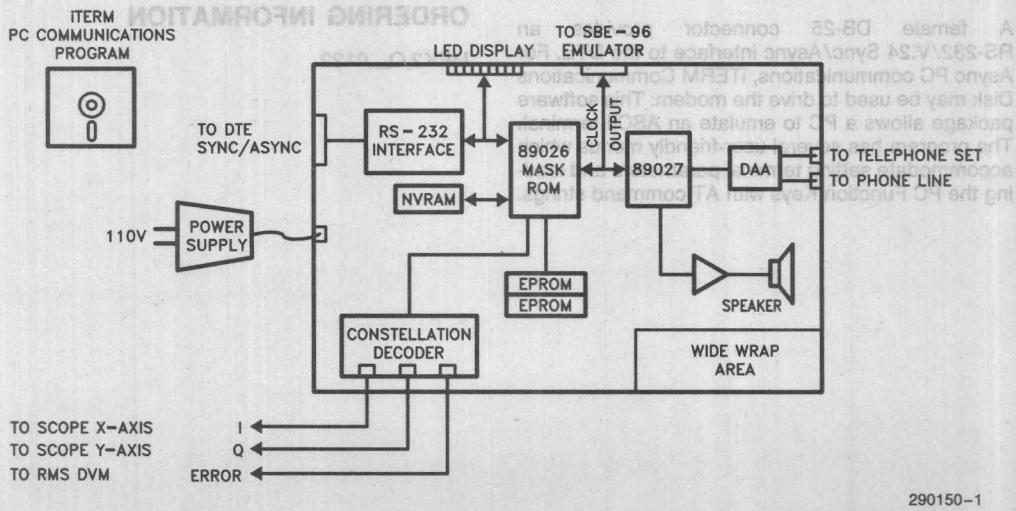
- **Single Board Evaluation Kit for the 89024 2400 bps Modem**
- **Recommended for Bit Error Rate Testing**
- **Intended as an 89024 Software Customization and Development Tool**
- **Comprehensive User's Manual**
- **PC Communications Package Included**
- **Includes Power Supply & User Wire-Wrap Area**

OVERVIEW

The 89024 MEK2 is a stand alone Sync/Async Intelligent Modem evaluation kit. This MULTI-BUS II form-factor circuit board is a fully assembled and functional modem that can be used for demonstrating the capabilities of the 89024 2400 bps Intelligent modem chip-set. It is also a versatile tool for evaluating the chip-set Bit Error Rate performance, as well as customizing the 89024 software.

The board is equipped with a power supply module, eliminating the need for a lab power supply, as well as a comprehensive user's manual and an Intel PC Communications Software package (iTERM).

4



290150-1

NOTE:
This product does not comply with FCC part 68 and part 15 requirements. It is intended for laboratory evaluation only.

Figure 1. 89024 MEK2 Block Diagram

March 1988
Order Number: 290150-002

SOFTWARE DEVELOPMENT

The 89024 MEK2 provides EPROM sockets for evaluating user developed software and provides the necessary clocking provisions to interface to an SBE-96 In-Circuit Emulator board.

For evaluation of modem signal quality, a constellation (eye pattern) decoder circuit has been included on-board. All hardware/software configurable features of the 89024 chip-set are strap configurable on this board.

HARDWARE OVERVIEW

The board provides a convenient vehicle for conducting Bit Error Rate tests. It is a good example of simple double-sided PCB layout that meet stringent modem noise requirements. The DAA interface supports Voice/Data Communications, as well as, 1A2 Key System A lead control. A loud-speaker and amplifier provide audible indication of call progress to the user. A series of LED indicators display the status of essential modem functions.

SERIAL INTERFACE

A female DB-25 connector provides an RS-232/V.24 Sync/Async interface to the DTE. For Async PC communications, iTERM Communications Disk may be used to drive the modem. This software package allows a PC to emulate an ASCII terminal. The program has several user-friendly menus which accommodate setting terminal parameters and loading the PC Function Keys with AT command strings.

DOCUMENTATION

Detailed information on the 89024 MEK2, iTERM and 89024 is provided. Following is a list of the enclosed documents.

1. 89024 MEK2 User's Guide
2. 89024 Reference Manual
3. 89024 Data Sheet
4. 89024 Firmware Evaluation Report

89024 Reference Manual

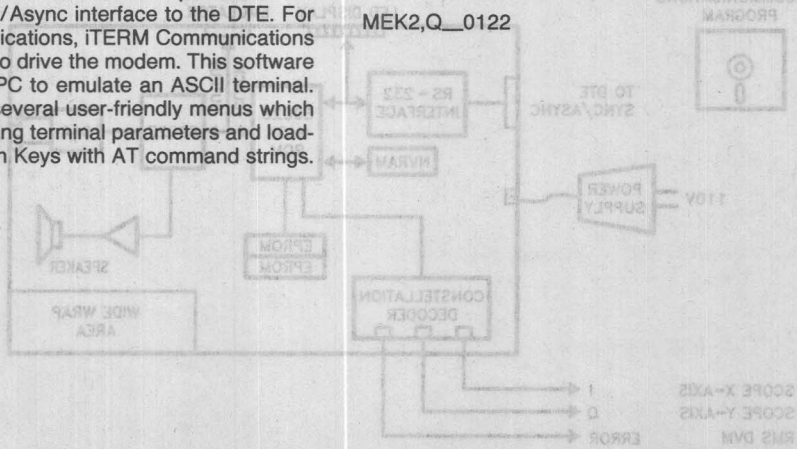
The 89024 Reference Manual details design information for the 89024 Modem Chip Set. It provides descriptions and specifications of the two chips comprising the 89024, the 89026 and the 89027. In addition, it describes the control interface between the two chips.

The reference manual also provides a full description of all the "AT" commands and S-registers supported by the 89024 Modem Chip Set.

Intel literature number: 296235-001

ORDERING INFORMATION

MEK2,Q__0122



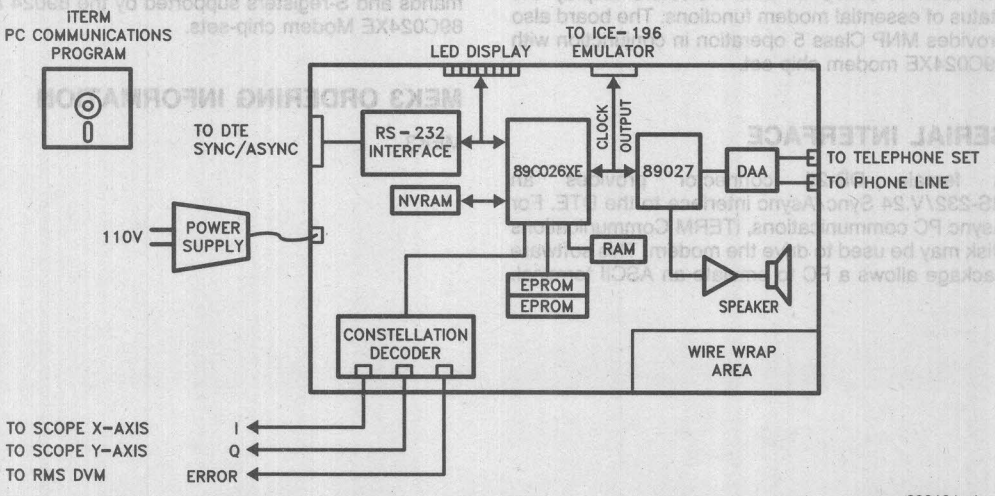
MEK3
MODEM EVALUATION KIT

- **Single Board Modem Evaluation Kit for the 89024, 89C024XE and MNP* Class 5 Firmware**
- **Onboard Constellation Decoder**
- **Software Customization and Development Platform for Intel Modem Chipsets**
- **Comprehensive User's Manual**
- **iTERM PC Communication Package**
- **User Wire-Wrap Area for H/W Customization**
- **Includes Power Supply**

OVERVIEW

The MEK3 is a modem evaluation kit for Intel's modem line of products. These include 89024 and 89C024XE-MNP Modems. This evaluation board comes assembled and tested as a functional modem. It is also designed to provide onboard prototyping area for purposes of customization. The packaged unit comes with the 89C024XE chip-set socketed on the board.

The board is equipped with a power supply module, eliminating the need for a lab power supply. A comprehensive user's manual and a copy of the Intel PC communications Software package iTERM is also included.



290191-1

NOTE:
This product is not FCC part 68 and part 15 approved. It is intended for laboratory evaluation only.

Figure 1. MEK3 Block Diagram

*MNP is a registered trademark of Microcom Incorporated.

October 1989
Order Number: 290191-002

HARDWARE OVERVIEW

The board provides a convenient vehicle for conducting Bit Error Rate tests. It is a good example of simple double-sided PCB layout that meet stringent modem noise requirements. The DAA interface supports Voice/Data Communications, as well as, 1A2 Key System A lead control. A loud-speaker and amplifier provide audible indication of call progress to the user. A series of LED indicators display the status of essential modem functions. The board also provides MNP Class 5 operation in conjunction with 89C024XE modem chip set.

SERIAL INTERFACE

A female DB-25 connector provides an RS-232/V.24 Sync/Async interface to the DTE. For Async PC communications, iTERM Communications Disk may be used to drive the modem. This software package allows a PC to emulate an ASCII terminal.

5. 89C024XE Data Sheet
4. 89024 Reference Manual
5. 89C024XE-MNP5 Modem design kit

Reference Manual

The Modem Reference Manual is available. It describes the 89024 and the 89C024XE chip-sets. It also provides a full description of all the "AT" commands and S-registers supported by the 89024 and 89C024XE Modem chip-sets.

MEK3 ORDERING INFORMATION

MEK3

May 1989

89024 Modem Customization for V.23 Data Transmission

BRIAN D. WALSH
APPLICATIONS ENGINEER
INTEL CORPORATION

Order Number: 292058-001

89024 MODEM CUSTOMIZATION FOR V.23 DATA TRANSMISSION

CONTENTS

	PAGE
INTRODUCTION	4-77
GENERAL DESCRIPTION	4-77
HARDWARE DESCRIPTION	4-77
Schematic Diagram	4-78
SOFTWARE DESCRIPTION	4-79
Decoding AT&A1 Command	4-79
Updating the Output Pins	4-80
V.23 Call Progress Monitor	4-82
V.23 Handshake	4-82
V.23 Data Mode	4-84
Assembling the Source Files	4-85
Linking the Object Files	4-85
Programming the EPROMs	4-85

INTRODUCTION

This application brief will illustrate the steps involved in customizing a modem application using the 89024 modem chip set. Specifically, it will show how one may add V.23 capability to an 89024 modem design as embodied in the MEK II (Intel Modem Evaluation Kit) running software version 3.2.

GENERAL DESCRIPTION

This design consists of using the 89026 processor to control a separate V.23 Data Pump IC (Texas Instruments TCM3105) to support V.23 modulation in addition to the currently supported V.22bis/V.22/V.21/Bell212/Bell103.

The modem is placed in V.23 mode using the "AT&A1" command and is returned to normal operation with the "AT&A0" command. The originating modem dials normally using "AT" commands and then 2 seconds after completion of dialing, the modem sends 75 bps V.23 carrier. The answering modem, upon detecting a ring signal, goes off hook and sends 1200 bps V.23 carrier. The originate modem sends data at 75 bps and receives data at 1200 bps, while the answer modem sends at 1200 bps and receives at 75 bps. Both respond to "escape" at 1200 bps and command mode is always at 1200/1200 bps. The V.23 transmit level is fixed. Backward channel CCITT circuits are not supported, data is always transmitted from pin 2 and received at pin 3.

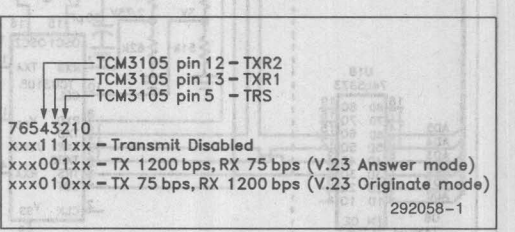
This application brief does not address the issues of V.25 calling tones or V.25 calling station identification.

HARDWARE DESCRIPTION

The MEK II is modified by adding a Texas Instruments TCM3105 FSK Modem IC. This Modem chip does not have an on-chip 4-wire to 2-wire hybrid circuit, so we use a dual op-amp MC1458 for this purpose. In order to control the TCM3105 we use 3 additional outputs of the 74LS373 latch that is already used to latch the /JS and AA signals from the microcontroller address/data bus. A 74LS157 2- to 1-line data selector is used to select the source of received data and the source of "energy detect" signal to the microcontroller.

V.23 Modem IC

The TCM3105 (U102) is a CMOS V.23 modem in a 16-pin package that consumes only 40 mW. It requires an external 4.4336 MHz crystal connected between pins 15 and 16 to derive timing. A resistor divider sets the carrier detect threshold by adjusting the voltage at pin 10. Bias distortion may be minimized by adjusting the voltage at pin 7. Pins 5, 13 and 12 together set the various modes of operation. These pins are connected to pins 6, 9 and 12 respectively of 74LS373 (U18) and are controlled through bits 2, 3 and 4 and executing a "STore" instruction to any even address of external memory (since this is the only external memory to be used). The modes of interest to us are:



74LS157 Data Selector

This IC is always enabled and the select signal is connected to the 6th output (bit 5) of the 74LS373 latch (U18). "SToring" a "0" to bit 5 of the latch selects "normal" mode of operation, while "SToring" a "1" to bit 5 selects V.23 mode. During "normal" mode, Receive Data (RXD) is routed from the 89026 microcontroller to the DTE and Energy Detect (ED) is routed from the 89027 AFE to the microcontroller. During V.23 mode RXD goes from the TCM3105 to the DTE and ED goes from the TCM3105 to the microcontroller. Transmit Data (TXD) is always connected from the DTE to both the 89026 and the TCM3105.

MC1458 Dual Op-Amp

This IC is configured as an active hybrid circuit, converting the 4-wire transmit and receive signals to 2-wire to drive the line transformer. The transmitted signal is also summed, but since only one of the transmitters will be active at a time, this will not be a problem. The 89027 has pin 10 tied low so as to disable the AFE's on-chip hybrid.

A schematic diagram of these changes is shown in Figure 1.

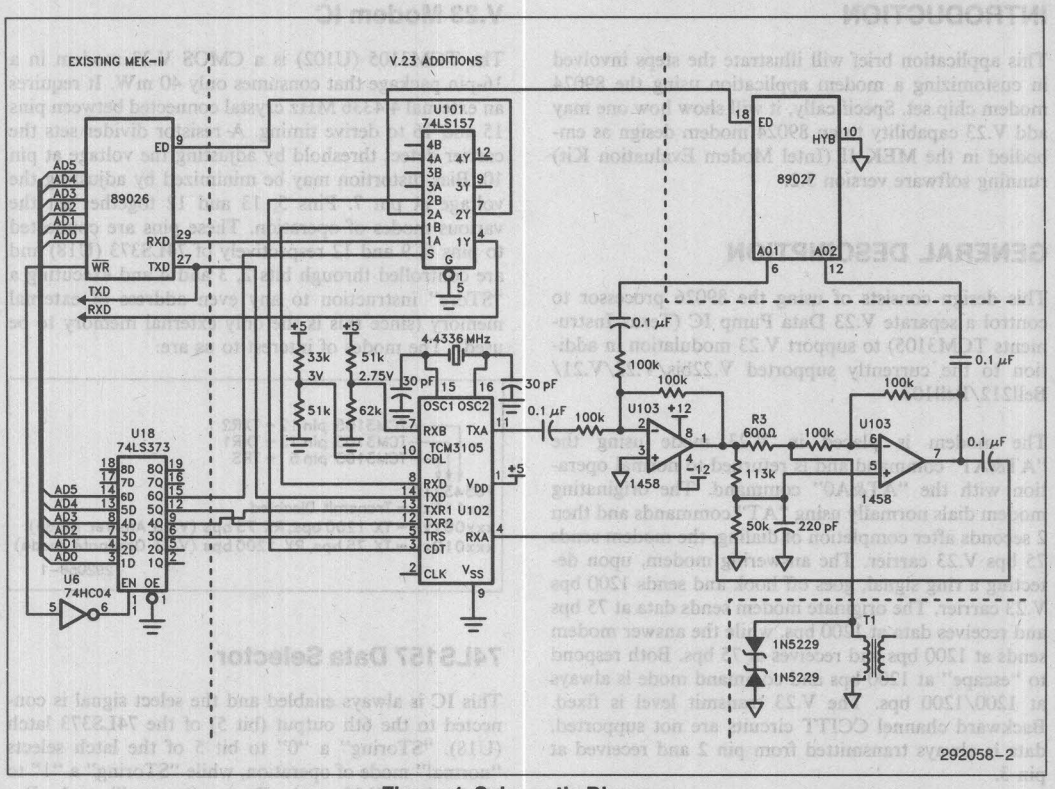


Figure 1. Schematic Diagram

SOFTWARE DESCRIPTION

We choose the "&A" command as one that is not currently used by major "AT" compatible modem vendors. We will use S23 bit 3 as the bit to indicate that V.23 mode has been selected, since this bit is unused in "AT" modems. "&A1" will cause S23 bit 3 to be set to a "1" and &A0 or just "&A" will cause it to be cleared. The modem software will examine this bit to determine whether V.23 mode has been selected.

Note that source code will always be written in capital letters and that the assembler ignores the rest of a line after a semi-colon (;). When giving modified source code I will usually "comment out" the original code by adding a semi-colon to the beginning of the line. This is an excellent practice to facilitate the documentation of changes.

By convention we name the source files: nmxxx.SRC (where n.m is the software version and xxx is the generic file name). Since we are using software version 3.2 the files that we will be changing are:

32AAD.SRC	register assignment definitions ($INCLUDEd with all source files)
32CMD.SRC	Command Decoder
32CPM.SRC	Call Progress Monitor routines
32HND.SRC	Handshake routines
32DATA.SRC	Data Mode routines

Decoding AT&A1 Command and Setting the S23 Bit
All of these changes will be done to the 32CMD.SRC file.

Since many commands simply modify S-register bits, we can take advantage of the "COMMON_REGISTER_OPERATIONS:" code by adding our command to the necessary tables and allowing it to be decoded as a register-modifying command.

Add as the last entry in TABLE_1:

```
DCB   (3 * 32) + (S23-S0)      ; AND_A_CMD
```

This will tell the common routine that this command affects bit 3 of S23. The table is set up so that it only occupies one byte per entry, with the bit number in the upper 3 bits and the register number in the lower 5 bits.

Add the command to the command list and the command vector table:

```
AND_CMDS:      DCB   "CJLPRSDG'
;              DCB   "MXFWZT',     0   ; was like this
               DCB   "MXAFWZT',    0   ; added &A command betw X and F
```

```
CMD_LU_TBL:    ....
               ....
;    DCB   AND_G_CMD-G1, AND_M_CMD-G1, AND_X_CMD-G1, AND_F_CMD-G2
     DCB   AND_G_CMD-G1, AND_M_CMD-G1, AND_X_CMD-G1, AND_A_CMD-G1
     DCB   AND_F_CMD-G2
```

The command vector table is the address offset of the command label from that of the first command (G1 EQU A_CMD). In the interests of saving space this offset table is only 1 byte per entry and so it has to be split into 2 groups as the range of addresses of command labels is more than 255 bytes. When modifying command code it is worth checking the list file to make sure that the CMD_LU_TBL: entries do not get bigger than 0FFH and wrap around through 0, causing those commands to branch to the wrong address.

Fix the branch vector calculator and the dial command offset calculator because the 1st group of commands are now 33 instead of 32:

```
GENERATE_BRANCH_VECTOR:
      ADD   TEMP_CMD_3, #G1              ; ADD OFFSET TO 1ST CMD GROUP
;     CMPB  TEMP_CMD_2, #32              ; FIRST 32 CMDS FIT IN
      CMPB  TEMP_CMD_2, #33              ; FIRST 33 CMDS FIT IN
```

```
D_R_CMD:
;     SUBB  CPM_CONTROL, TEMP_CMD_2, #36
      SUBB  CPM_CONTROL, TEMP_CMD_2, #37
```

Add the command label with the rest of the register modifying commands:

```
Y_CMD:
AND_A_CMD:    ; added &A command for V.23 operation
AND_C_CMD:
```

Updating the Output Pins to Control the TCM3105 and Data Selector

The IO_CONTROL: section of code in file 32CMD.SRC runs all the time and could be considered the "background routine". This is where the RS232 leads are updated, the health of the other routines is checked and the 74LS373 latch (U18) is written and is thus an appropriate place for the TCM3105 chip and the Data Selector (Data Mux) to be updated.

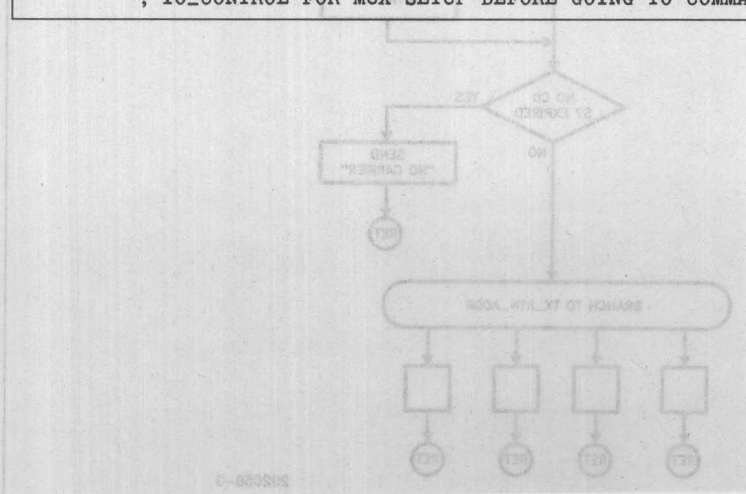
Add the following code after END_JS_UPDATE:

```
END_JS_UPDATE:
V_23_UPDATE:
  ANDB TEMP_CMD_1, #11011111B       ; MUX TO NON-V.23 POSN
  ORB  TEMP_CMD_1, #00011100B       ; SET V.23 CHIP OFF
  JBC  S23,  3, END_V_23_UPDATE     ; JMP IF NOT IN V.23 MODE

  JBC  CNTRL_C, 1, END_V_23_UPDATE ; JMP IF NOT IN HND OR DATA MODE
  ANDB TEMP_CMD_1, #11100111B       ; SET V.23 CHIP TO ANS MODE
  ORB  TEMP_CMD_1, #00000100B

  JBC  S14, 7, NOT_ORIG_MODE        ; JMP IF S REG SET TO ANS MODE
  ANDB TEMP_CMD_1, #11101011B       ; SET V.23 CHIP TO ORIG MODE
  ORB  TEMP_CMD_1, #00001000B

NOT_ORIG_MODE:
  JBC  CNTRL_C, 0, END_V_23_UPDATE ; JMP IF NOT IN DATA MODE
  JBS  CNTRL_C, 2, END_V_23_UPDATE ; JMP IF CMD FUNCTS ENABLED
  ORB  TEMP_CMD_1, #00100000B       ; DATA MODE, SO MUX TO V.23 POSN

END_V_23_UPDATE:
```

The next instruction in the source code STores the contents of TEMP_CMD_1 to PORT3, and so updates the Data Mux.

In order to ensure that the Data Mux gets set before the "OK" message is sent when entering the on-line escape state (response to "+++"), add a line of code after the three "ORB" instructions:

```
VALID_ESCAPE_SEQUENCE:
  ORB  CNTRL_F, #00010000B        ; ENABLE ESCAPE STATE
  ORB  CNTRL_C, #00000100B        ; ENABLE CMD FUNCTIONS
  ORB  MSG_RQST, #00100000B       ; SEND "OK" MESSAGE WITH MSG RQST
  JBS  S23, 3, ESCAPE_DETECT_END ; TRICK TO FORCE 1 MORE PASS THRU
       ; IO_CONTROL FOR MUX SETUP BEFORE GOING TO COMMAND DECODER
```



After a dial command is executed by the Command routine, it will activate the Call Progress routines.

The V.23 Call Progress Monitor Routines

The 32CPM.SRC routines check for call progress signals on the phone line and also for answer tone from the remote answering modem. Since a V.23 modem will answer with a 1300 Hz tone (1200 bps mark frequency), the AFE receive filter must be set to V.22 answer mode so as to pass this frequency to the energy detect circuitry.

Add three lines of code at the label SET_ANSWER_CONT:

```
SET_ANSWER_CONT:
  ANDB CPM_FLAG, #11101111B ; FLAG ANSWER PROCESSING FOR HOUSEKEEPING

  JBC  S23,  3, SET_ANSWER_CONT_1 ; IF V23 MODE THEN
  LDB  AFE_BYTE3, #01000000B      ; SET FILTER TO QAM ANS FOR
SET_ANSWER_CONT_1:                ; 1300Hz CARRIER DETECTION

  SJMP SIGNAL_MONITOR_INIT
```

The CPM routines will hand over control to the Handshake routines which we need to modify for V.23 handshake.

The V.23 Handshake Routines

The Handshake mode 32HND.SRC is entered for the first time after successful completion of the Call Progress routines. The first time that HANDSHAKE_MODE: is called, it goes through the Initialization code before the main routine is executed, thereafter the Initialization is skipped. The Main routine is entered at a rate of 600 times per second or more and consists of checking for Energy Detect and then branching to the routine address saved in TX_RTN_ADDR. The logical flow of the handshaking is controlled by changing the contents of TX_RTN_ADDR to the address of the routine to be executed the next time Handshake is called.

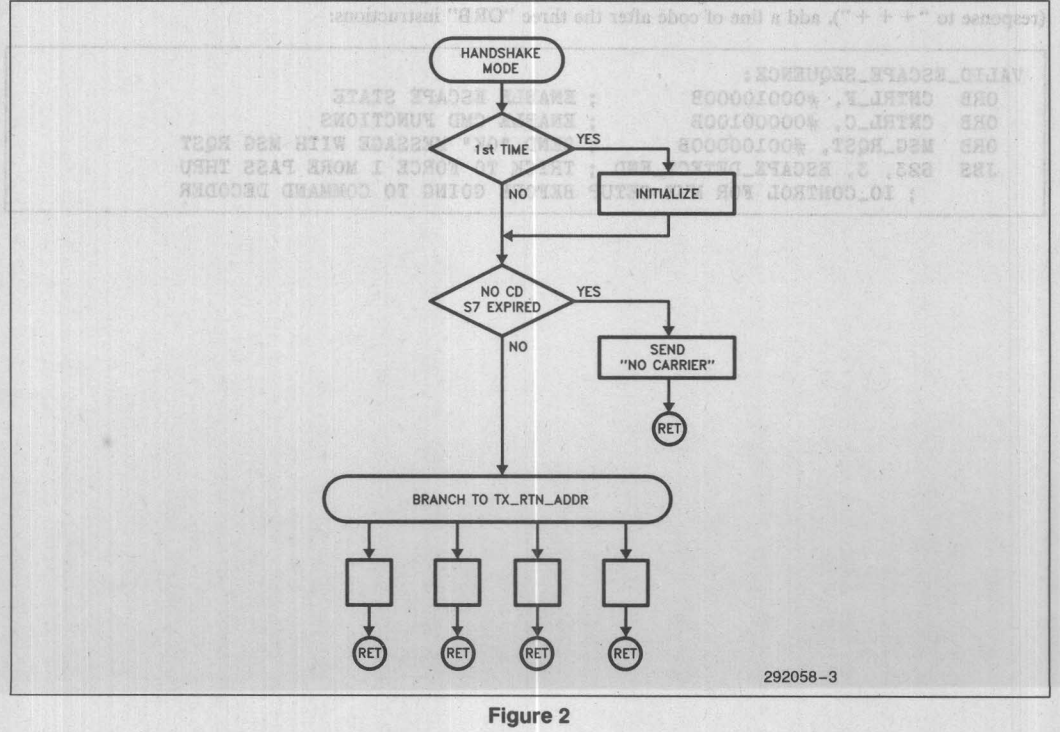


Figure 2

The Initialization required for V.23 consists of starting the S7 wait-for-carrier timer, starting a 2 second timer and loading a return address for the next time the routine executes. The following lines of source code are added (identified by "V23" at the start of the comment field) to the Handshake Initialization:

```
HANDSHAKE_INIT:
  ANDB MODE_STATUS, #10111111B          ; CLEAR INIT FLAG
                                        ; V23
  JBC  S23,  3,  NOT_V23_INIT           ; V23
V23_HND_INIT:                           ; V23
  ADDB S7_TIMER, TIME_BASE_SECOND, S7   ; V23 INIT S7 DCD TIMER
  ADDB TX_TIMER, TIME_BASE_100MS, #20D  ; V23 INIT 2 SEC TIMER
  LD   TX_RTN_ADDR,   #V23_HND_WAIT     ; V23
  SJMP HND_INIT_END                     ; V23
                                        ; V23
NOT_V23_INIT:                           ; V23
```

After the initialization code is executed once, the software will keep branching to V23_HND_WAIT: until the 2-second timer has expired, then it will initiate a "CONNECT" message. While the Connect message is being sent, the software will branch to V23_HND_MESSAGE:, then it will set up the Data mode and thereafter the Data Mode will be called instead of the Handshake mode.

```
V23_HND_WAIT:                           ; V23
  CMPB TIME_BASE_100MS, TX_TIMER        ; V23 TIMER EXPIRED YET?
  JNE  V23_HND_END                      ; V23
V23_HND_MESSAGE_INIT:                   ; V23
  LDB  MESSAGE_REQUEST, #00100001B      ; V23 START CONNECT MESSAGE
  LD   TX_RTN_ADDR,   #V23_HND_MESSAGE  ; V23
  SJMP V23_HND_END                      ; V23
                                        ; V23
V23_HND_MESSAGE:                        ; V23
  JBS  MESSAGE_REQUEST, 5, V23_HND_END  ; V23 MESSAGE SENT YET?
V23_HND_MESSAGE_END:                    ; V23
                                        ; V23
  ANDB COPY_PORT4, #10111111B           ; V23 DCD HIGH AFTER CONNECT
V23_SET_UP_DATA_MODE:                   ; V23
; This is where we need to set up for going to data mode
  ORB   CNTRL_C, #00000011B             ; V23 GO TO DATA MODE
  ANDB  AFE_BYTE4, #00111111B           ; V23 TXMITTER OFF, AFE OFF
  CLRB  DM_FLAGS                        ; V23 CLEAR FLAGS FOR DM
  ORB   MODE_STATUS, #10000000B         ; V23 INIT DATA MODE
V23_HND_END:                            ; V23
  LJMP  HANDSHAKE_MODE_END              ; V23
                                        ; V23
```

4

V.23 Data Mode

The modifications required in the Data Mode consist of checking for V23 mode and skipping past:

- Initialization
- Send space disconnect (twice)
- Receive space disconnect
- Loss of carrier disconnect
- Retrain request
- Test mode

```
DATA_MODE_INIT:
; DM FLAGS ALREADY CLEARED IN HANDSHAKE MODE
  ANDB MODE_STATUS,#7FH              ; CLEAR INITIALIZE FLAG
  JBS S23, 3, DATA_MODE_INIT_END     ; IF V23 THEN INIT DONE
```

```
DISCONNECT_INIT:
  ANDB DM_FLAGS, #11111101B          ; CLEAR DISCONNECT INIT FLAG
  JBS  S23,  3,  HANG_UP             ; V23 FORGET SPACE DISCONNECT
```

```
SEND_SPACE:
  JBC  S21, 7, HANG_UP               ; IF BREAK_DISCONNECT DISABLED
  JBS  S23, 3,  HANG_UP              ; V23 FORGET BREAK
```

```
CHECK_DISCONNECT:                    ; CHECK FOR LONG SPACE DISC
CHECK_BREAK:
  JBS  S23, 3,  SET_BREAK_TIME       ; V23 FORGET BREAK
```

```
CHECK_CARRIER_LOSS:
  JBS  PORT0, 7, CHECK_CARRIER_LOSS_END ; SKIP IF ED IS HIGH
  ORB  DM_FLAGS, #01000000B             ; SET CDLOSS FLAG
  ADDB EDOFF_TIME,TIME_BASE_100MS,S10 ; CDOFF THRESHOLD IN REGISTER
  INCB EDOFF_TIME                  ; PUT AN OFFSET IN TIME FOR PROPER
                                   ; OPERATION DURING TM EXIT
  JBS S23, 3, CARRIER_LOSS_END     ; ALL DONE IF V23 MODE
```

```
QAM_RETRAIN:
  JBS S23, 3, SJMP_CHECK_TEST_MODE   ; SKIP RETRAIN IF V23 MODE
```

```
CHECK_S16_STATUS:
; EXAMINE S16 REGISTER FOR ANY TEST MODES AND SET FLAG
  JBS S23, 3, CHECK_S16_STATUS_END   ; SKIP RETRAIN IF V23 MODE
```

Assembling the Source Files
The source files can be assembled by issuing the following commands at the DOS prompt:

```
ASM96 32CMD.SRC
ASM96 32CPM.SRC
ASM96 32HND.SRC
ASM96 32DATA.SRC
```

Linking the Object Files
Link the object files by issuing the following command at the DOS prompt:

```
RL96 32HND.OBJ, 32INIT.OBJ, 32CMD.OBJ, 32CPM.OBJ, 32DATA.OBJ,
     32SOFT.OBJ, 32HSI.OBJ, 32HSO.OBJ, 32RX.OBJ TO 32ATR
```

Programming the EPROMs
After the code has been linked and located, the code must be split into low and high byte segments for programming into EPROMS. The following IPPS session illustrates that process (IPPS prompts are not shown):

```
IPPS                          ; invoke IPPS
I 80                          ; initialize file format
FORMAT 32ATR                  ; filename resulting from linking
3                             ; logical unit is byte
2                             ; input file is in words (2 bytes)
1                             ; output file is in bytes
0 to  32ATR.LO                ; low order bytes to one file
1 to  32ATR.HI                ; high order bytes to another
<enter>                       ; press "enter" to exit formatting
                              ;
                              ; the following assumes that an
                              ; INTEL PiUP 201A programmer is
                              ; connected to the PC
                              ;
TYPE                          ; display available EPROM types
27128                         ; specify EPROM type
                              ; insert blank EPROM into programmer
COPY 32ATR.LO TO PROM         ; copy low byte file to prom
                              ; insert blank EPROM into programmer
COPY 32ATR.HI TO PROM         ; copy low byte file to prom
EX                            ; exit IPPS
```

Custom routines can now be tested by placing EPROMS into target hardware.

REFERENCES

1. "FSK Modems: TCM3105 Designers Information" from Telecommunications Circuits Data Book, 1986. By Texas Instruments.
2. MEKII 89024 Enhanced Modem Evaluation Kit Users Manual, 1987. By Intel Corp.
3. 89024 Modem Reference Manual, 1987. By Intel Corp.
4. Developing MCS-96 Applications Using the SBE-96. Application Note AP-273 (Order Number 280249-001). By Intel Corp.

Assembling the Source Files

The source files can be assembled by issuing the following commands at the DOS prompt:

```
ASM96 32CMD.SRC
ASM96 32CPM.SRC
ASM96 32RMD.SRC
ASM96 32DATA.SRC
```

Linking the Object Files

Link the object files by issuing the following command at the DOS prompt:

```
RL96 32RMD.OBJ, 32INIT.OBJ, 32CMD.OBJ, 32CPM.OBJ, 32DATA.OBJ,
     32SOFT.OBJ, 32HSI.OBJ, 32HSO.OBJ, 32RX.OBJ TO 32ATR
```

Programming the EPROMs

After the code has been linked and located, the code must be split into low and high byte segments for programming into EPROMS. The following IPPS session illustrates that process (IPPS prompts are not shown):

```
IPPS                                    ; invoke IPPS
I 80                                    ; initialize file format
FORMAT 32ATR                            ; filename resulting from linking
3                                       ; logical unit is byte
2                                       ; input file is in words (2 bytes)
1                                       ; output file is in bytes
0 to 32ATR.LO                           ; low order bytes to one file
1 to 32ATR.HI                           ; high order bytes to another
<enter>                                 ; press "enter" to exit formatting
                                        ;
                                        ; the following assumes that an
                                        ; INTEL PIUP 20IA programmer is
                                        ; connected to the PC
                                        ;
TYPE                                    ; display available EPROM types
SPEED                                   ; specify EPROM type
                                        ; insert blank EPROM into programmer
COPY 32ATR.LO TO PROM                   ; copy low byte file to prom
                                        ; insert blank EPROM into programmer
COPY 32ATR.HI TO PROM                   ; copy low byte file to prom
EX                                      ; exit IPPS
```

Custom routines can now be tested by placing EPROMS into target hardware.

REFERENCES

1. "FSK Modems: TCM3105 Designers Information" from Telecommunications Circuits Data Book, 1986. By Texas Instruments.
2. MEKII 80924 Enhanced Modem Evaluation Kit Users Manual, 1987. By Intel Corp.
3. 80924 Modem Reference Manual, 1987. By Intel Corp.
4. Developing MCS-96 Applications Using the SBE-96. Application Note AP-273 (Order Number 280249-001). By Intel Corp.

PCM Codec/Filter and Combo

5

5 PCM Codec/Filter and Combo

2910A
PCM CODEC - μLAW
8-BIT COMPANDED A/D AND D/A CONVERTER

- **Per Channel, Single Chip Codec**
- **CCITT G711 and G712 Compatible, ATT T1 Compatible with 8th Bit Signaling**
- **Microcomputer Interface with On-Chip Timeslot Computation**
- **Simple Direct Mode Interface When Fixed Timeslots are Used**
- **±5% Power Supplies: +12V, +5V, −5V**
- **78dB Dynamic Range, with Resolution Equivalent to 12-Bit Linear Conversion Around Zero**
- **Precision On-Chip Voltage Reference**
- **Low Power Consumption 230 mW Typ. Standby Power 33mW Typ.**
- **Fabricated with Reliable N-Channel MOS Process**

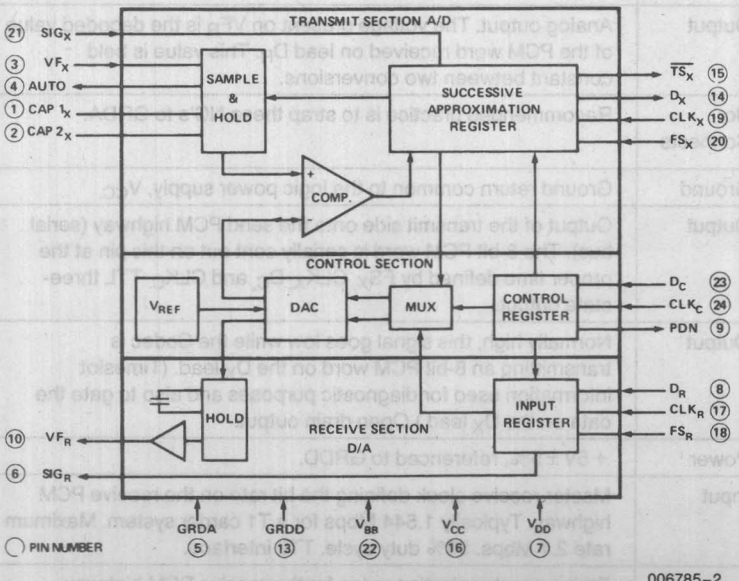
The Intel 2910A is a fully integrated PCM (Pulse Code Modulation) Codec (Coder-Decoder), fabricated with N-channel silicon gate technology. The high density of integration allows the sample and hold circuits, the digital-to-analog converter, the comparator and the successive approximation register to be integrated on the same chip, along with the logic necessary to interface a full duplex PCM link and provide in-band signaling.

The primary applications are in telephone systems:

- Transmission —T1 Carrier
- Switching —Digital PBX's and Central Office Switching Systems
- Concentration —Subscriber Carrier/Concentrators

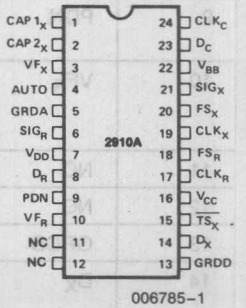
The wide dynamic range of the 2910A (78dB) and the minimal conversion time (80μsec minimum) make it an ideal product for other applications, like:

- Date Acquisition • Telemetry • Secure Communications Systems • Signal Processing Systems



006785-2

Figure 1. Block Diagram



006785-1

Figure 2. Pin Configuration

CAP 1_X, CAP 2_X	Holding Capacitor
VF_X	Analog Input
VF_R	Analog Output
D_R, D_C, SIG_X	Digital Input
SIG_R, D_X, $\overline{TS}_X$	Digital Output
CLK_C, CLK_X, CLK_R	Clock Input
FS_X, FS_R	Frame Sync Input
AUTO	Auto Zero Output
V_{BB}	Power (−5V)
V_{CC}	Power (+5V)
V_{DD}	Power (+12V)
PDN	Power Down
GRDA	Analog Ground
GRDD	Digital Ground
NC	No Connect

Figure 3. Pin Names

5

November 1986
Order Number: 006785-002

Pin Description

Pin No.	Symbol	Function	Description
1	$CAP1_X$	Hold	Connections for the transmit holding capacitor. Refer to Applications section.
2	$CAP2_X$		
3	VF_X	Input	Analog input to be encoded into a PCM word. The signal on this lead is sampled at the same rate as the transmit frame synchronization pulse FS_X, and the sample value is held in the external capacitor connected to the $CAP1_X$ and $CAP2_X$ leads until the encoding process is completed.
4	AUTO	Output	Most significant bit of the encoded PCM word (+5V for negative, −5V for positive inputs). Refer to the Codec Applications section.
5	GRDA	Ground	Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.
6	SIG_R	Output	Signaling output. SIG_R is updated with the 8th bit of the receive PCM word on signaling frames, and is latched between two signaling frames. TTL interface.
7	V_{DD}	Power	+12V ± 5%; referenced to GRDA.
8	D_R	Input	Receive PCM highway (serial bus) interface. The Codec serially receives a PCM word (8 bits) through this lead at the proper time defined by FS_R, CLK_R, D_C, and CLK_C.
9	PDN	Output	Active high when Codec is in the power down state. Open drain output.
10	VF_R	Output	Analog output. The voltage present on VF_R is the decoded value of the PCM word received on lead D_R. This value is held constant between two conversions.
11	NC	No Connects	Recommended practice is to strap these NC's to GRDA.
12	NC		
13	GRDD	Ground	Ground return common to the logic power supply, V_{CC}.
14	D_X	Output	Output of the transmit side onto the send PCM highway (serial bus). The 8-bit PCM word is serially sent out on this pin at the proper time defined by FS_X, CLK_X, D_C, and CLK_C. TTL three-state output.
15	$\overline{TS}_X$	Output	Normally high, this signal goes low while the Codec is transmitting an 8-bit PCM word on the D_X lead. (Timeslot information used for diagnostic purposes and also to gate the data on the D_X lead.) Open drain output.
16	V_{CC}	Power	+5V ± 5%, referenced to GRDD.
17	CLK_R	Input	Master receive clock defining the bit rate on the receive PCM highway. Typically 1.544 Mbps for a T1 carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.
18	FS_R	Input	Frame synchronization pulse for the receive PCM highway. Resets the on-chip timeslot counter for the receive side. Maximum repetition rate 12 KHz. Also used to differentiate between non-signaling frames and signaling frames on the receive side. TTL interface.

Pin Description (Continued)

Pin No.	Symbol	Function	Description
19	CLK_X	Input	Master transmit clock defining the bit rate on the transmit PCM highway. Typically 1.544 Mbps for a T1 carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.
20	FS_X	Input	Frame synchronization pulse for the transmit PCM highway. Resets the on-chip timeslot counter for the transmit side. Maximum repetition rate 12 KHz. Also used to differentiate between non-signaling frames and signaling frames on the transmit side. TTL interface.
21	SIG_X	Input	Signaling input. This digital input is transmitted as the 8th bit of the PCM word on the D_X lead, on signaling frames. TTL interface.
22	V_{BB}	Power	$-5V \pm 5\%$, referenced to GRDA.
23	D_C	Input	Data input to program the Codec for the chosen mode of operation. Becomes an active low chip select when CLK_C is tied to V_{CC}. TTL interface.
24	CLK_C	Input	Clock input to clock in the data on the D_C lead when the timeslot assignment feature is used; tied to V_{CC} to disable this feature. TTL interface.

FUNCTIONAL DESCRIPTION

The 2910A PCM Codec provides the analog-to-digital and the digital-to-analog conversions necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system.

In a typical telephone system the Codec is used between the PCM highways and the channel filters.

The Codec provides two major functions:

- Encoding and decoding of analog signals (voice and call progress tones)
- Encoding and decoding of the signaling and supervision information

On a non-signaling frame, the Codec encodes the incoming analog signal at the frame rate (FS_X) into an 8-bit PCM word which is sent out on the D_X lead at the proper time. Similarly, the Codec fetches an 8-bit PCM word from the receive highway (D_R lead) and decodes an analog value which will remain constant on lead VF_R until the next receive frame. Transmit and receive frames are independent. They can be asynchronous (transmission) or synchronous (switching) with each other.

For channel associated signaling, the Codec transmit side will encode the incoming analog signal as previously described and substitute the signal present on lead SIG_X for the least significant bit of the encoded PCM word. Similarly, on a receive signaling frame, the Codec will decode the 7 most significant bits according to the CCITT G733 recommendation and will output the least significant bit value on the SIG_R lead until the next signaling frame. Signaling frames on the send and receive sides are independent of each other, and are selected by a double-width frame sync pulse on the appropriate channel.

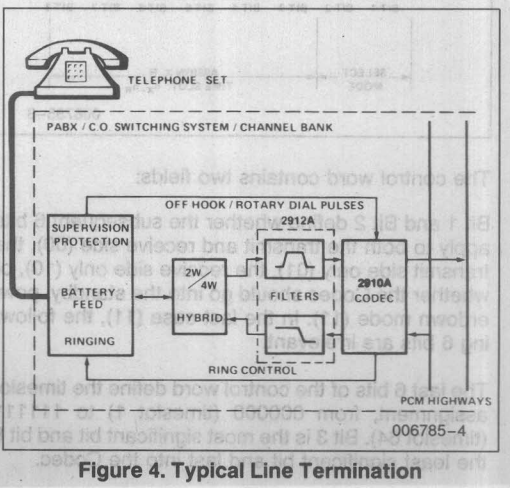


Figure 4. Typical Line Termination

The 2910A Codec is intended to be used on line and trunk terminations. The call progress tones (dial tone, busy tone, ring-back tone, re-order tone), and the prerecorded announcements, can be sent through the voice-path; digital signaling (off hook and disconnect supervision, rotary dial pulses, ring control) is sent through the signaling path.

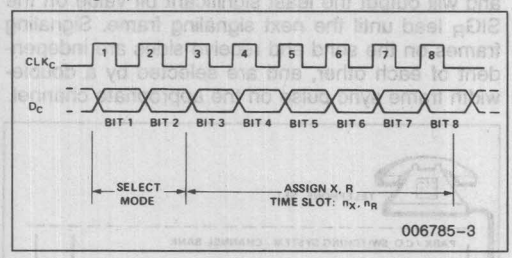
Circuitry is provided within the Codec to internally define the transmit and receive timeslots. In small systems this may eliminate the need for any external timeslot exchange; in large systems it provides one level of concentration. This feature can be bypassed and discrete timeslots sent to each Codec within a system.

In the power-down mode, most functions of the Codec are directly disabled to reduce power dissipation to a minimum.

CODEC OPERATION

Codec Control

The operation of the 2910A is defined by serially loading an 8-bit word through the D_C lead (data) and the CLK_C lead (clock). The loading is asynchronous with the other operations of the Codec, and takes place whenever transitions occur on the CLK_C lead. The D_C input is loaded in during the trailing edge of the CLK_C input.



The control word contains two fields:

Bit 1 and Bit 2 define whether the subsequent 6 bits apply to both the transmit and receive side (00), the transmit side only (01), the receive side only (10), or whether the Codec should go into the standby, powerdown mode (11). In the last case (11), the following 6 bits are irrelevant.

The last 6 bits of the control word define the timeslot assignment, from 000000 (timeslot 1) to 111111 (timeslot 64). Bit 3 is the most significant bit and bit 8 the least significant bit and last into the Codec.

Bit 1	Bit 2	Mode
0	0	X & R
0	1	X
1	0	R
1	1	Standby

Bit						
3	4	5	6	7	8	Timeslot
0	0	0	0	0	0	1
0	0	0	0	0	1	2
		•				•
		•				•
		•				•
		•				•
1	1	1	1	1	1	64

The Codec will retain the control word (or words) until a new word is loaded in or until power is lost. This feature permits dynamic allocation of timeslots for switching applications.

Microcomputer Control Mode

In the microcomputer mode, each Codec performs its own timeslot computation independently for the transmit and receive channels by counting clock pulses (CLK_X and CLK_R). All Codecs tied to the same data bus receive identical framing pulses (FS_X and FS_R). The framing pulses reset the on-chip timeslot counters every frame; hence the timeslot counters of all devices are synchronized. Each Codec is programmed via CLK_C and D_C for the desired transmit and receive timeslots according to the description in the Codec Control Section. All Codecs tied to the same D_R bus will, in general, have different receive timeslots, although that is not a device requirement. There may be separate busses for transmit and receive or all Codecs may transmit and receive over the same bus, in which case the transmit and receive channels must be synchronous ($CLK_X = CLK_R$). There are no other restrictions on timeslot assignments; a device may have the same transmit and receive timeslot even if a single bus is used.

There are several requirements for using the CLK_C-D_C interface in the microcomputer mode.

1) A complete timeslot assignment, consisting of eight negative transitions of CLK_C, must be made in less than one frame period. The assignment

can overlap a framing pulse so long as all 8 control bits are clocked in within a total span of 125 μs (for an 8 KHz frame rate). CLK_C must be left at a TTL low level when not assigning a timeslot.

2) A dead period of two frames must always be observed between successive timeslot assignments. The two frame delay is measured from the rising edge of the first CLK_C transition of the previous timeslot assigned.

3) When the device is in the power-down state (Standby), the following three-step sequence must be followed to power-up the codec to avoid contention on the transmit PCM highway.

   a) Assign a dummy transmit timeslot. The dummy should be at least two timeslots greater than the maximum valid system timeslot (usually 24 or 32). For example, in a 24 timeslot system, the dummy could be any timeslot between 26 and 64. This will power-up the transmit side, but prevent any spurious D_X or $\overline{TS}_X$ outputs.

   b) Two frames later, assign the desired transmit timeslot.

   c) Two frames later assign the desired receive timeslot.

4) Initialization sequence: The device contains an on-chip power-on clear function which guarantees that with proper sequencing of the supplies (V_{CC} or V_{DD} on last), the device will initialize with no timeslot assigned to either the transmit or receive channel. After a supply failure or whenever the supplies are applied, it is recommended that either power down assignment be made first, or the first timeslot assignment be a transmit timeslot or a transmit/receive timeslot. The consequence of making a receive timeslot assignment first, after supply application, is that the transmit channel will assume timeslot 1, potentially producing bus contention.

5) Transmit only/receive only operation is permitted provided that a power down assignment is made first. Otherwise, special circuits which use only one channel should be physically disconnected from the unused bus; this allows a timeslot to be made to an unused channel without consequence.

6) Both frame synchronizing pulses (FS_X, FS_R) must be active at all times after power on clear (after power supplies are turned on). This requirement must be met during powerdown and receive only or transmit only operation, as well as during normal transmit and receive operation.

Example of Microcomputer Control Mode:

The two words 01000001 and 10000010 have been loaded into the Codec. The transmit side is now programmed for timeslot 2 and the receive side for timeslot 3. The Codec will output a PCM word on the transmit PCM highway (bus) during the timeslot 2 of the transmit frame, and will fetch a PCM word from the receive PCM highway during timeslot 3.

5

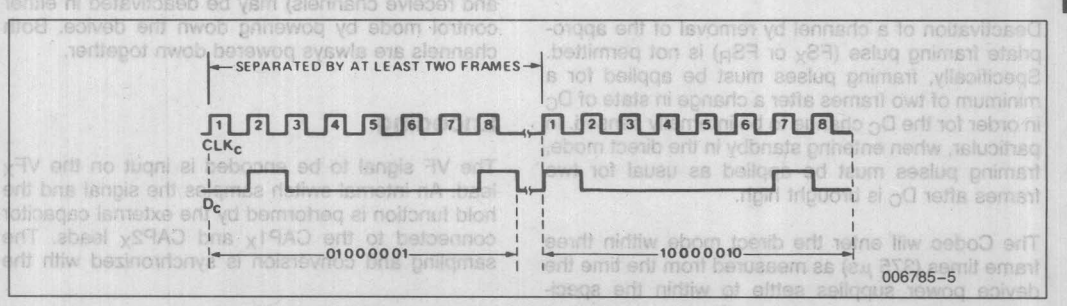


Figure 5. Microcomputer Mode Programming Example

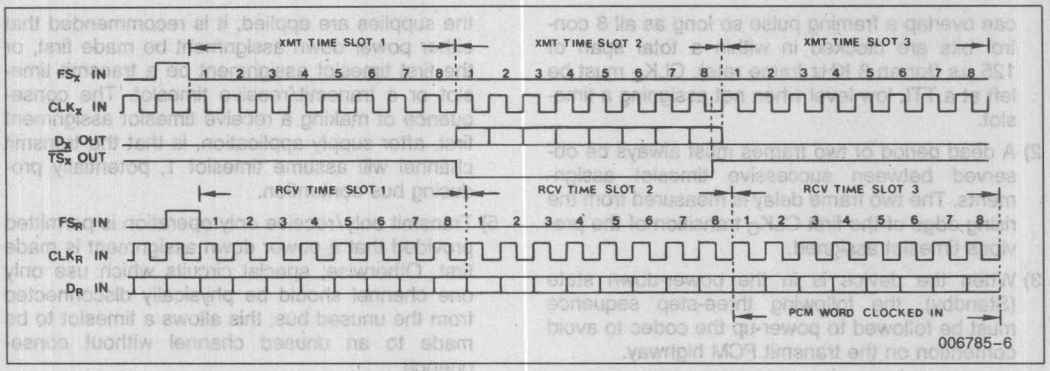


Figure 6. Microcomputer Mode PCM Highway Example

In this example the Codec interface to the PCM highway then functions as shown above. (FS_X and FS_R may be asynchronous.)

Direct Control Mode

The direct mode of operation will be selected when the CLK_C pin is strapped to the +5 volt supply (V_{CC}). In this mode, the D_C pin is an active low chip select. In other words, when D_C is low, the device transmits and receives in the timeslots which follow the appropriate framing pulses. With D_C high the device is in the power down state. Even though CLK_C characteristics are simpler for the 2910A it will operate properly when plugged into a 2910 board.

Deactivation of a channel by removal of the appropriate framing pulse (FS_X or FS_R) is not permitted. Specifically, framing pulses must be applied for a minimum of two frames after a change in state of D_C in order for the D_C change to be internally sensed. In particular, when entering standby in the direct mode, framing pulses must be applied as usual for two frames after D_C is brought high.

The Codec will enter the direct mode within three frame times (375 μs) as measured from the time the device power supplies settle to within the specified limits. This assumes that CLK_C is tied to V_{CC} and that all clocks are available at the time the supplies have settled.

General Control Requirements

All bit and frame clocks should be applied whenever the device is active. In particular, an unused channel cannot be deactivated by removal of its associated frame or bit clock while the other channel of the same device remains active.

A single channel cannot be deactivated except by physical disconnection of the data lead (D_X or D_R) from the system data bus. A device (both transmit and receive channels) may be deactivated in either control mode by powering down the device. Both channels are always powered down together.

Encoding

The VF signal to be encoded is input on the VF_X lead. An internal switch samples the signal and the hold function is performed by the external capacitor connected to the $CAP1_X$ and $CAP2_X$ leads. The sampling and conversion is synchronized with the

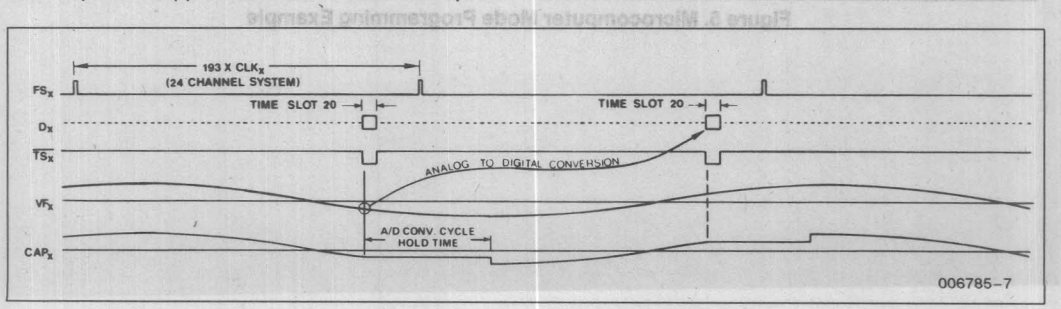


Figure 7. Transmit Encoding

transmit timeslot. The PCM word is then output on the D_X lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately ±2.2 volts RMS (±3.1 volts peak).

Decoding

The PCM word is fetched by the D_R lead from the PCM highway at the proper timeslot occurrence. The decoded value is held on an internal sample and hold capacitor. The buffered non-return to zero output signal on the VF_R lead has a dynamic range of approximately ±2.2 volts RMS (±3.1 volts peak).

Signaling

The duration of the FS_X and FS_R pulses defines whether a frame is an information frame or a signaling frame:

- A frame synchronization pulse which is a full clock period in duration (CLK_X period for FS_X, CLK_R period for FS_R) designates a non-signaling frame.
- A frame synchronization pulse which is two full clock periods in duration (two CLK_X periods for FS_X, two CLK_R periods for FS_R) designates a signaling frame.

On the encoding side, when the FS_X pulse is widened, the 8th bit of the PCM word will be replaced by the value on the SIG_X input at the time when the 8th bit is output on the D_X lead.

On the decoding side, when the FS_R pulse is widened, the 8th bit of the PCM word is detected and transmitted on the SIG_R lead. That output is latched until the next receiving signaling frame.

The remaining 7 bits are decoded according to the value given in the CCITT G733 recommendation. The SIG_R lead is reset to a TTL low level whenever the Codec is in the power-down state.

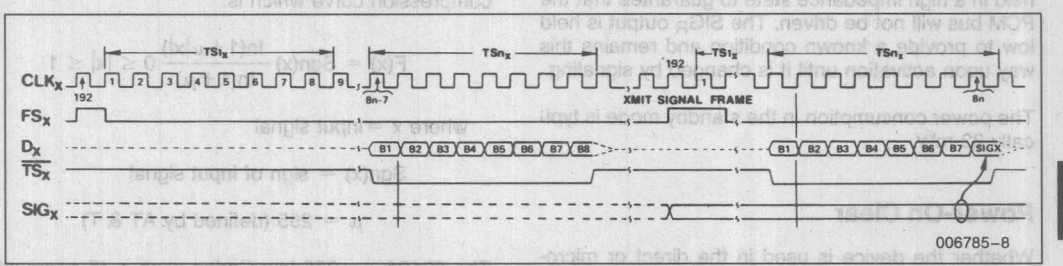


Figure 8. Transmit 8th Bit Signaling

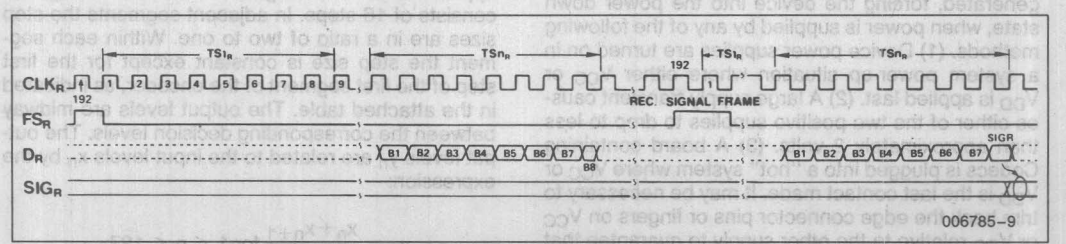


Figure 9. Receive 8th Bit Signaling

5

T1 Framing

The Codec will accept the standard D3/D4 framing format of 193 clock pulses per frame (equivalent to CLK_X, CLK_R of 1.544 Mb/s). However, the 193rd bit may be blanked (equivalent to CLK_X, CLK_R of 1.536 Mb/s) if desired.

Standby Mode—Power Down

To minimize power consumption and heat dissipation a standby mode is provided where all Codec functions are disabled except for D_C and CLK_C leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (D_C) with a "1" in bits 1 and 2 locations. In the direct mode when D_C is brought high, the all "1's" control word is internally transferred to the control register, invoking the standby condition.

While in the standby mode, the D_X output is actively held in a high impedance state to guarantee that the PCM bus will not be driven. The SIG_R output is held low to provide a known condition and remains this way upon activation until it is changed by signaling.

The power consumption in the standby mode is typically 33 mW.

Power-On Clear

Whether the device is used in the direct or microcomputer mode, an internal reset (power-on clear) is generated, forcing the device into the power down state, when power is supplied by any of the following methods. (1) Device power supplies are turned on in a system power-up situation where either V_{CC} or V_{DD} is applied last. (2) A large supply transient causes either of the two positive supplies to drop to less than approximately 2 volts. (3) A board containing Codecs is plugged into a "hot" system where V_{CC} or V_{DD} is the last contact made. It may be necessary to trim back the edge connector pins or fingers on V_{CC} or V_{DD} relative to the other supply to guarantee that the power-on clear will operate properly when a board is plugged into a "hot" system. Furthermore, the Codec will inhibit activity on $\overline{TS}x$ and D_X during the application of power supplies.

The device is also tolerant of transients in the negative supply (V_{BB}) so long as V_{BB} remains more negative than −3.5 volts. V_{BB} transients which exceed this level should be detected and followed by a system reinitialization.

Precision Voltage Reference for the D/A Converter

The voltage reference is generated on the chip and is calibrated during the manufacturing process. The technique uses the difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature stable and bias stable reference voltage.

A gain setting op amp, programmed during manufacturing, "trims" the reference voltage source to the final precision voltage reference value provided to the D/A converter. The precision voltage reference determines the initial gain and dynamic range characteristics described in the A.C. Transmission Specification section.

μ-Law Conversion

μ-law represents a particular implementation of a piece-wise linear approximation to a logarithmic compression curve which is:

$$F(x) = \mathrm{Sgn}(x)\frac{\ln(1+\mu|x|)}{\ln(1+\mu)} \quad 0 \le |x| \le 1$$

where x = input signal

Sgn(x) = sign of input signal

μ = 255 (defined by AT & T)

The 2910A μ = 255 law Codec uses a 15 segment approximation to the logarithmic law. Each segment consists of 16 steps. In adjacent segments the step sizes are in a ratio of two to one. Within each segment the step size is constant except for the first step of the first segment of the encoder, as indicated in the attached table. The output levels are midway between the corresponding decision levels. The output levels y_n are related to the input levels x_n by the expression:

$$y_n = \frac{x_n + x_{n+1}}{2} \text{ for } 1 \le n \le 127$$

$$y_0 = x_0 = 0 \text{ for } n = 0$$

These relationships are implicit in the following table.

Theoretical μ-Law—Positive Input Values (for Negative Input Values, Invert Bit 1)

1	2	3	4	5	6	7	8
Segment Number	No. of Steps x Step Size	Value at Segment End Points	Decision Value Number n	Decision Value x_n(1)	PCM Word(3) MSB 1 2 3 4 5 6 7 8 LSB	Normalized Value at Decoder Output y_n(4)	Decoder Output Value Number
		8159(5)	(128)	(8159)			
					1 0 0 0 0 0 0 0	8031	127
			127	7903			
8	16 x 256				(see Note 2)		
			113	4319			
					1 0 0 0 1 1 1 1	4191	112
		4063	112	4063			
7	16 x 128				(see Note 2)		
			97	2143			
					1 0 0 1 1 1 1 1	2079	96
		2015	96	2015			
6	16 x 64				(see Note 2)		
			81	1055			
					1 0 1 0 1 1 1 1	1023	80
		991	80	991			
5	16 x 32				(see Note 2)		
			65	511			
					1 0 1 1 1 1 1 1	495	64
		479	64	479			
4	16 x 16				(see Note 2)		
			49	239			
					1 1 0 0 1 1 1 1	231	48
		223	48	223			
3	16 x 8				(see Note 2)		
			33	103			
					1 1 0 1 1 1 1 1	99	32
		95	32	95			
2	16 x 4				(see Note 2)		
			17	35			
					1 1 1 0 1 1 1 1	33	16
		31	16	31			
	15 x 2				(see Note 2)		
			2	3			
1					1 1 1 1 1 1 1 0	2	1
↓	1 x 1		1	1			
					1 1 1 1 1 1 1 1		
			0	0		0	0

NOTES:
1. 8159 normalized value units correspond to the value of the on-chip voltage reference.
2. The PCM word corresponding to positive input values between two successive decision values numbered n and n+1 (see column 4) is (255−n) expressed as a binary number.
3. The PCM word on the highways is the same as the one shown in column 6.
4. The voltage output on the VF_R lead is equal to the normalized value given in the table, augmented by an offset. The offset value is approximately 15 mV.
5. x_{128} is a virtual decision value.

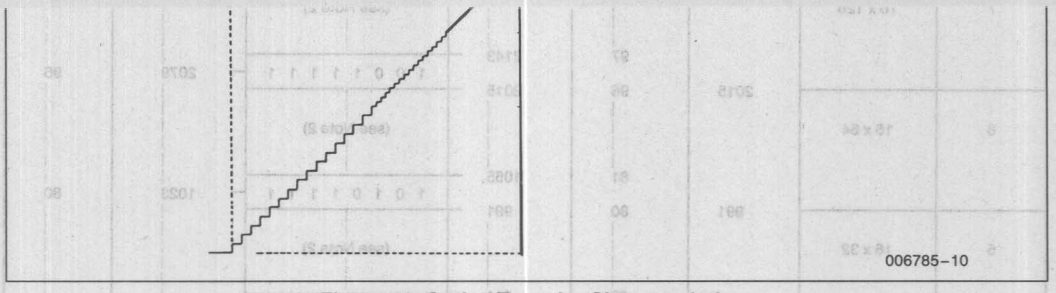


Figure 10. Codec Transfer Characteristic

During signaling frames, a 7-bit transfer characteristic is implemented in the decoder. This characteristic is derived from the decoder values in the attached table by assuming a value of "1" for the LSB (8th bit) and shifting the decoder transfer characteristics one half-step away from the origin. For example, the maximum decoder output level for signaling frames has normalized value 7903, whereas it has value 8031 in normal (non-signaling) frames.

APPLICATIONS

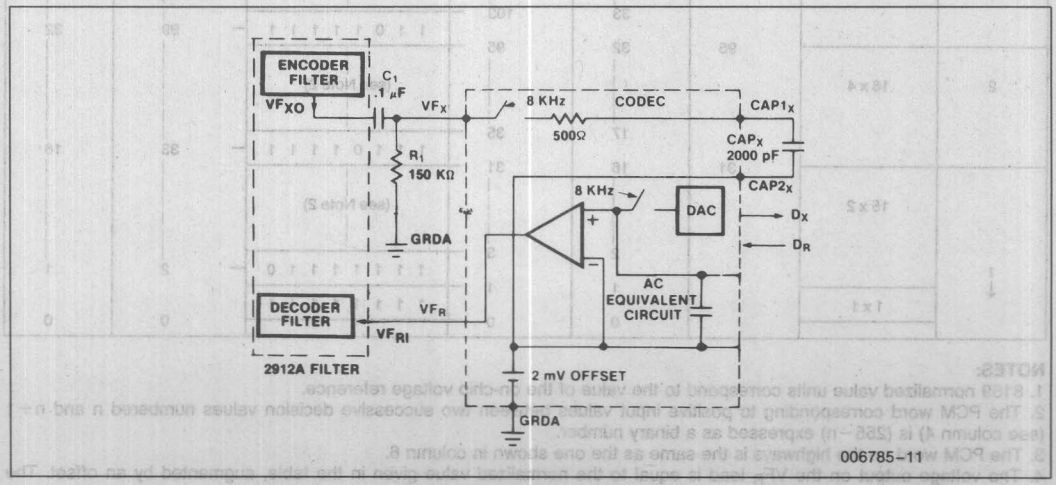


Figure 11. Circuit Interface—without External Auto Zero

Holding Capacitor

For an 8 KHz sampling system the transmit holding capacitor CAP_X should be 2000 pF ±20%.

Auto Zero

The 2910A contains a transparent on-chip auto zero plus a device pin for implementing a sign-bit driven external auto zero feedback loop. The on-chip auto zero reduces the input offset voltage of the encoder (VF_X) to less than 3 mV. For most telephony applications, this input offset is perfectly acceptable, since it insures the encoder is biased in the lower 25% of the first segment.

Where lower input offset is required the external auto zero loop may be used to bias the encoder exactly at the zero crossing point. The consequence of the external auto zero loop, aside from extra components, is the addition of the dithering auto-zero signal to the input signal, resulting in slightly higher idle channel noise (approximately 2dB) than when the external loop is not used. Consequently, where the application permits, it is recommended that the external auto zero loop not be used. When not used, the AUTO pin should float.

The circuit interface with auto zero drawing shows a possible connection between the VF_X and AUTO leads with the recommended values of $C_1 = 0.3\ \mu F$, $R_1 = 150\ K\Omega$, $R_2 = 330\ K\Omega$, and $R_3 = 470\ K\Omega$.

Filters Interface

The filters may be interfaced as shown in the circuit interface diagrams. Note that the output pulse stream is of the non-return-to-zero type and hence requires the (sin x)/x correction provided by the 2912A filter.

D_X Buffering

For higher drive capability or increased system reliability it may be desirable that the D_X output of a group of Codecs be buffered from the system PCM bus with an external three-state or open collector buffers. A buffer can be enabled with the appropriate Codec generated $\overline{TS}x$ signal or signals. $\overline{TS}x$ signal may also be used to activate external zero code suppression logic, since the occurrence of an active state of any $\overline{TS}x$ implies the existence of PCM voice bits (as opposed to transparent data bits) on the bus.

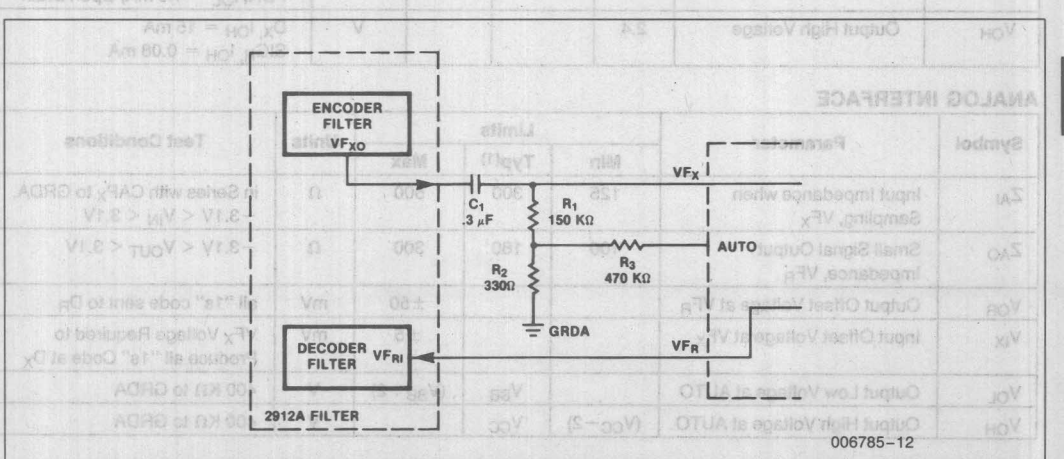


Figure 12. Circuit Interface—with External Auto Zero

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +80°C

Storage Temperature ..... −65°C to +150°C

All Input or Output Voltages with Respect to V_{BB} ..... −0.3V to +20V

V_{CC}, V_{DD}, GRDD, and GRDA with Respect to V_{BB} ..... −0.3V to +20V

Power Dissipation 1.35W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

$T_A = 0°C$ to $+70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified

DIGITAL INTERFACE

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ(1)	Max		
I_{IL}	Low Level Input Current			10	μA	$V_{IN} < V_{IL}$
I_{IH}	High Level Input Current			10	μA	$V_{IN} > V_{IH}$
V_{IL}	Input Low Voltage			0.6	V	
V_{IH}	Input High Voltage	2.0			V	
V_{OL}	Output Low Voltage			0.4	V	D_X, I_{OL} = 4.0 mA SIG_R, I_{OL} = 0.5 mA $\overline{TSx}$, I_{OL} = 3.2 mA, Open Drain PDN, I_{OL} = 1.6 mA, Open Drain
V_{OH}	Output High Voltage	2.4			V	D_X, I_{OH} = 15 mA SIG_R, I_{OH} = 0.08 mA

ANALOG INTERFACE

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ(1)	Max		
Z_{AI}	Input Impedance when Sampling, VF_X	125	300	500	Ω	in Series with CAP_X to GRDA, $-3.1V < V_{IN} < 3.1V$
Z_{AO}	Small Signal Output Impedance, VF_R	100	180	300	Ω	$-3.1V < V_{OUT} < 3.1V$
V_{OR}	Output Offset Voltage at VF_R			±50	mV	all "1s" code sent to D_R
V_{IX}	Input Offset Voltage at VF_X			±5	mV	VF_X Voltage Required to Produce all "1s" Code at D_X
V_{OL}	Output Low Voltage at AUTO		V_{BB}	(V_{BB} + 2)	V	400 KΩ to GRDA
V_{OH}	Output High Voltage at AUTO	(V_{CC} − 2)	V_{CC}		V	400 KΩ to GRDA

POWER DISSIPATION

Symbol	Parameter	Limits			Units	Test Conditions
		Min	Typ(1)	Max		
I_{DDO}	Standby Current		0.7	1.1	mA	Auto Output = Open Clock Frequency = 2.048 MHz
I_{CCO}	Standby Current		4	7.0	mA	
I_{BBO}	Standby Current		1	2.5	mA	
I_{DDI}	Operating Current		11	16	mA	
I_{CCI}	Operating Current		13	21	mA	
I_{BBI}	Operating Current		4	6.0	mA	

NOTE:

1. Typical values are for T_A = 25°C and nominal power supply values.

A.C. CHARACTERISTICS

$T_A = 0°C$ to $+70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified

TRANSMISSION

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
S/D	Signal/Tone Distortion Ratio,					VF_X = 1.02 KHz, Sinusoid
	C-Message Weighted	36			dB	-30 dBm0 $\leq VF_X \leq 0$ dBm0
	Half Channel	30			dB	-40 dBm0 $\leq VF_X < -30$ dBm0
	(See Figure 1)	27			dB	-45 dBm0 $\leq VF_X < -40$ dBm0
ΔG						VF_X = 1.02 KHz, Sinusoid
	Gain Tracking Deviation		±0.25	±0.30	dB	-37 dBm0 $\leq VF_X \leq +3$ dBm0
	Half Channel(2)		±0.60	±0.70	dB	-50 dBm0 $\leq VF_X < -37$ dBm0
	Reference Level 0 dBm0		±1.5	±1.8	dB	-55 dBm0 $\leq VF_X < -50$ dBm0
ΔG_V	ΔG Variation with Supplies		±0.0002	±0.0004	dB/mV	-37 dBm0 $\leq VF_X \leq +3$ dBm0
	Half Channel		±0.0004	±0.0008	dB/mV	-50 dBm0 $\leq VF_X < -37$ dBm0
ΔG_T	ΔG Variation with Temperature		±0.001	±0.002	dB/°C	-37 dBm0 $\leq VF_X \leq +3$ dBm0
	Half Channel		±0.002	±0.005	dB/°C	-50 dBm0 $\leq VF_X < -37$ dBm0
N_{IC1}	Idle Channel Noise, C-Message Weighted		2	7	dBrnc0	No Signaling(3)
N_{IC2}	Idle Channel Noise, C-Message Weighted		10	13	dBrnc0	with 6th and 12th Frame Signaling(3)
N_{IC3}	Idle Channel Noise, C-Message Weighted		14	18	dBrnc0	with 1 KHz Sign Bit Toggle
HD	Harmonic Distortion (2nd or 3rd)		-48	-44	dB	VF_X = 1.02 KHz, 0 dBm0; Measured at Decoder Output VF_R
IMD	Intermodulation Distortion					4-Tone Stimulus in Accordance
	2nd Order			-45	dB	with BSTR PUB 41009
	3rd Order			-55	dB	

NOTES:
1. Typical values are for T_A = 25°C and nominal supply values.
2. Measured in one direction, either decoder or encoder and an ideal device, at 23°C, nominal supplies.
3. If the external auto-zero is used N_{IC1} has a typical value of 8 dBrnc0 and N_{IC2} has a typical value of 13 dBrnc0.
4. D_R of Device Under Test (D.U.T.) driven with repetitive digital word sequence specified in CCITT recommendation G.711. Measurement made at VF_R output.
5. With the D.C. method the positive and negative clipping levels are measured and A_{IR} is calculated. With the A.C. method a sinusoidal input signal to VF_X is used where A_{IR} is measured directly.

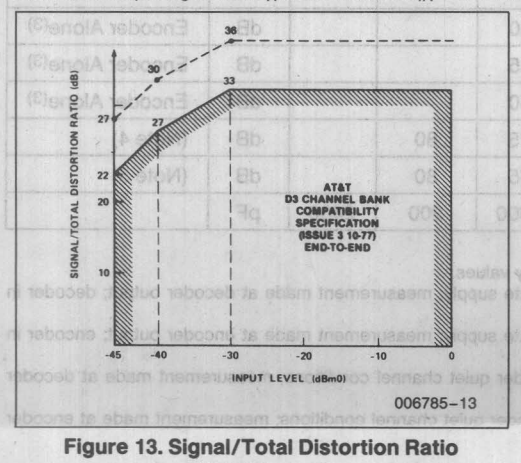


Figure 13. Signal/Total Distortion Ratio (Half-Channel)

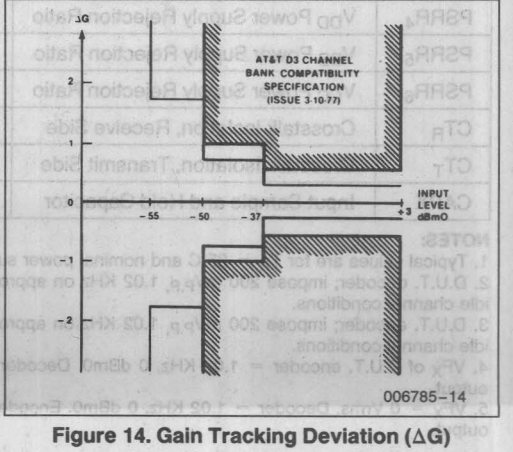


Figure 14. Gain Tracking Deviation (ΔG) (Half-Channel)

A.C. CHARACTERISTICS

$T_A = 0°C$ to $+70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified (Continued)

GAIN AND DYNAMIC RANGE

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
DmW	Digital Milliwatt Response	5.53	5.63	5.73	dBm	23°C, Nominal Supplies(4)
DmW_T	DmW_O Variation with Temperature		−0.001	−0.002	dB/°C	Relative to 23°C(4)
DmW_S	DmW_O Variation with Supplies			±0.07	dB	Supplies ±5%(4)
A_{IR}	Input Dynamic Range	2.17	2.20	2.23	V_{RMS}	Using D.C. and A. C. Tests(5) 23°C, Nominal Supplies
A_{IRT}	Input Dynamic Range with Temperature			−0.5	mV_{RMS}/°C	Relative to 23°C
A_{IRS}	Input Dynamic Range with Supplies			±18	mV_{RMS}	Supplies ±5%
A_{OR}	Output Dynamic Range, VF_R	2.13	2.16	2.19	V_{RMS}	23°C, Nominal Supplies
A_{ORT}	A_{OR} Variation with Temperature			−0.5	mV_{RMS}/°C	Relative to 23°C
A_{ORS}	A_{OR} Variation with Supplies			±18	mV_{RMS}	Supplies ±5%

SUPPLY REJECTION AND CROSSTALK

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
$PSRR_1$	V_{DD} Power Supply Rejection Ratio	45			dB	Decoder Alone(2)
$PSRR_2$	V_{BB} Power Supply Rejection Ratio	35			dB	Decoder Alone(2)
$PSRR_3$	V_{CC} Power Supply Rejection Ratio	50			dB	Decoder Alone(2)
$PSRR_4$	V_{DD} Power Supply Rejection Ratio	50			dB	Encoder Alone(3)
$PSRR_5$	V_{BB} Power Supply Rejection Ratio	45			dB	Encoder Alone(3)
$PSRR_6$	V_{CC} Power Supply Rejection Ratio	50			dB	Encoder Alone(3)
CT_R	Crosstalk Isolation, Receive Side	75	80		dB	(Note 4)
CT_T	Crosstalk Isolation, Transmit Side	75	80		dB	(Note 5)
CAPX	Input Sample and Hold Capacitor	1600	200	2400	pF	

NOTES:
1. Typical values are for $T_A = 25°C$ and nominal power supply values.
2. D.U.T. decoder; impose 200 $mV_{P,P}$, 1.02 KHz on appropriate supply; measurement made at decoder output; decoder in idle channel conditions.
3. D.U.T. encoder; impose 200 $mV_{P,P}$, 1.02 KHz on appropriate supply; measurement made at encoder output; encoder in idle channel conditions.
4. VF_X of D.U.T. encoder = 1.02 KHz, 0 dBm0. Decoder under quiet channel conditions; measurement made at decoder output.
5. VF_X = 0 Vrms. Decoder = 1.02 KHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.

A.C. CHARACTERISTIC—TIMING SPECIFICATION(1)

$T_A = 0°C$ to $+70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = +5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, GRDA = 0V, GRDD = 0V, unless otherwise specified

CLOCK SECTION

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
t_{CY}	Clock Period	485		ns	CLK_X, CLK_R (2.048 MHz Systems), CLK_C
t_r, t_f	Clock Rise and Fall Time	0	30	ns	CLK_X, CLK_R, CLK_C
t_{CLK}	Clock Pulse Width	215		ns	CLK_X, CLK_R, CLK_C
t_{CDC}	Clock Duty Cycle ($t_{CLK} \div t_{CY}$)	45	55	%	CLK_X, CLK_R

TRANSMIT SECTION

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
t_{VFX}	Analog Input Conversion	20		Timeslot	from Leading Edge of Transmit Timeslot (2)
t_{DZX}	Data Enabled on TS Entry	50	180	ns	$0 < C_{LOAD} < 100$ pF
t_{DHX}	Data Hold Time	80	230	ns	$0 < C_{LOAD} < 100$ pF
t_{HZX}	Data Float on TS Exit	75	245	ns	$C_{LOAD} = 0$
t_{SON}	Timeslot X to Enable	30	220	ns	$0 < C_{LOAD} < 100$pF
t_{SOFF}	Timeslot X to Disable	70	225	ns	$C_{LOAD} = 0$
t_{SS}	Signal Setup Time	0		ns	Relative to Bit 7 Falling Edge
t_{SH}	Signal Hold Time	100		ns	Relative to Bit 8 Falling Edge
t_{FSD}	Frame Sync Delay	15	150	ns	

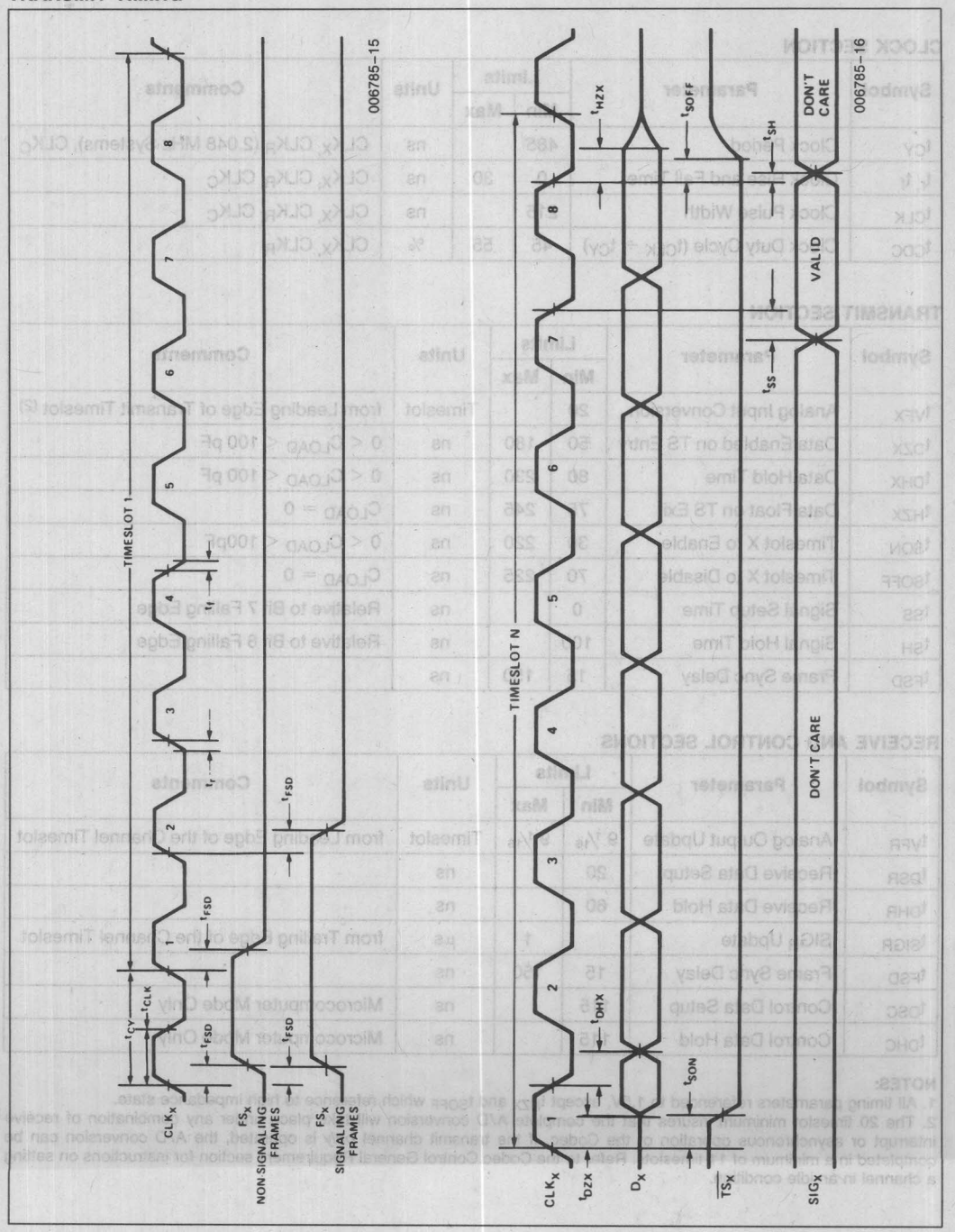
RECEIVE AND CONTROL SECTIONS

Symbol	Parameter	Limits		Units	Comments
		Min	Max		
t_{VFR}	Analog Output Update	9 1/16	9 1/16	Timeslot	from Leading Edge of the Channel Timeslot
t_{DSR}	Receive Data Setup	20		ns	
t_{DHR}	Receive Data Hold	60		ns	
t_{SIGR}	SIG_R Update		1	μs	from Trailing Edge of the Channel Timeslot
t_{FSD}	Frame Sync Delay	15	150	ns	
t_{DSC}	Control Data Setup	115		ns	Microcomputer Mode Only
t_{DHC}	Control Data Hold	115		ns	Microcomputer Mode Only

NOTES:
1. All timing parameters referenced to 1.5V, except t_{HZX} and t_{SOFF} which reference to high impedance state.
2. The 20 timeslot minimum insures that the complete A/D conversion will take place under any combination of receive interrupt or asynchronous operation of the Codec. If the transmit channel *only* is operated, the A/D conversion can be completed in a minimum of 11 timeslots. Refer to the Codec Control General Requirement section for instructions on setting a channel in an idle condition.

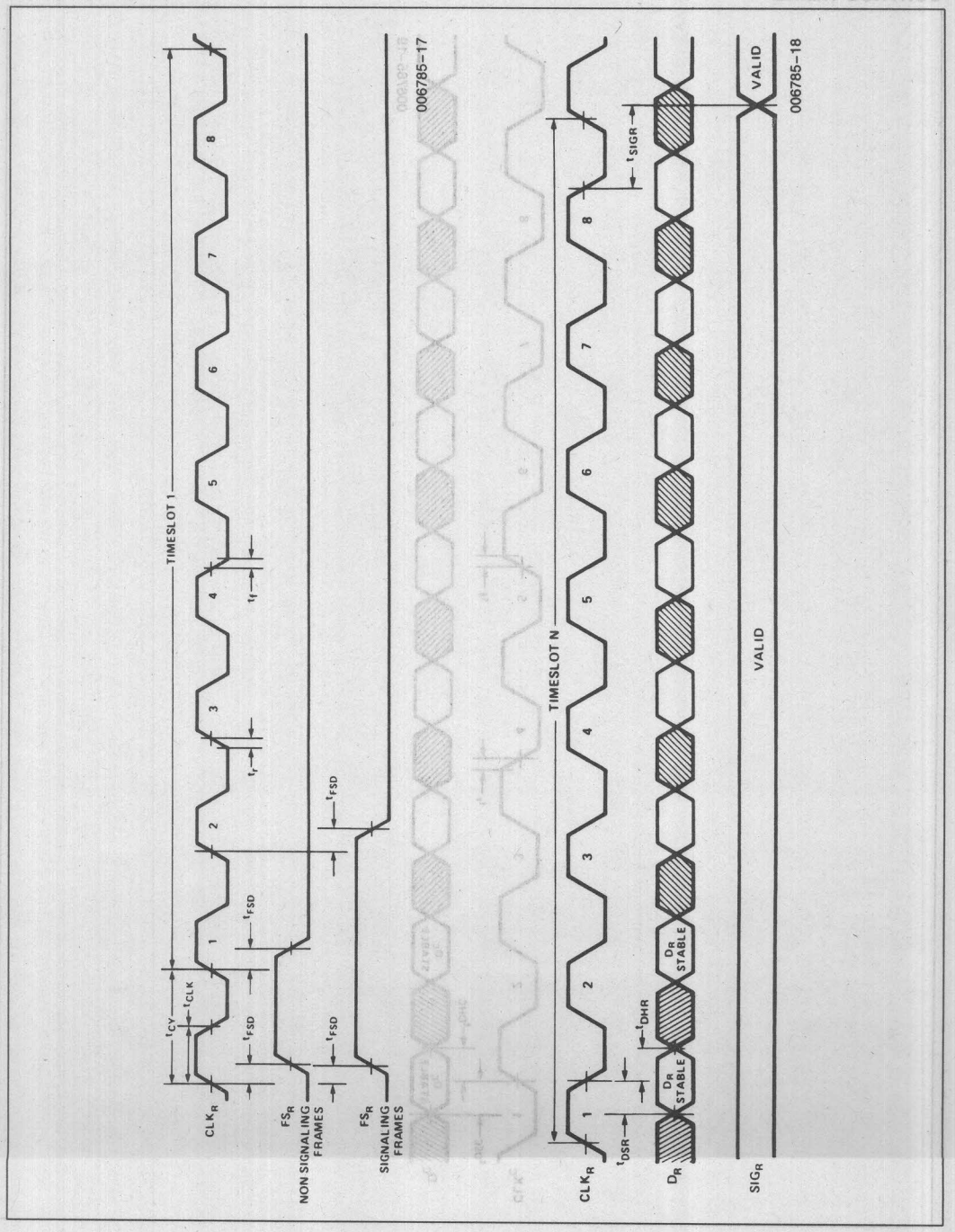
TIMING WAVEFORMS

TRANSMIT TIMING



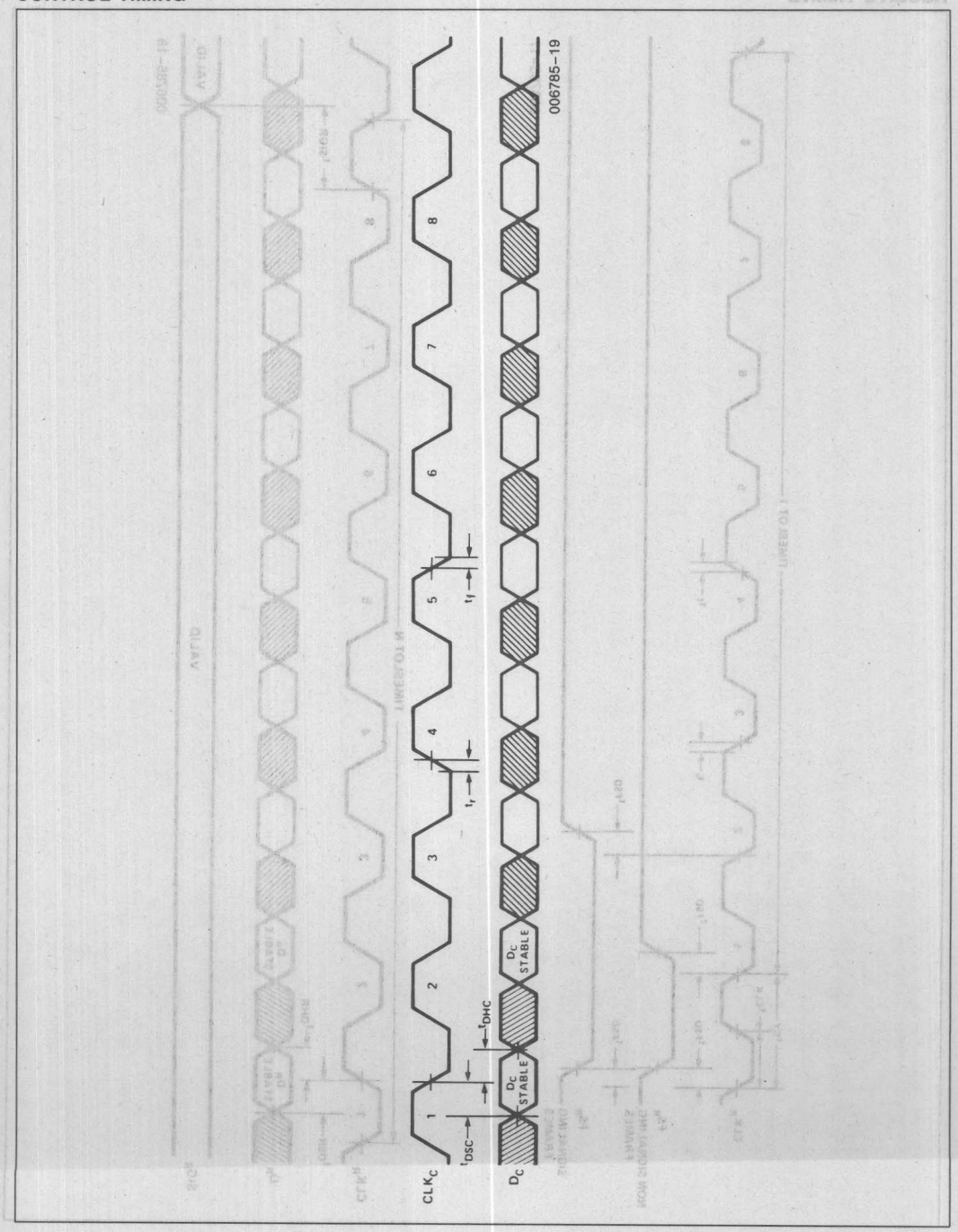
TIMING WAVEFORMS (Continued)

RECEIVE TIMING



TIMING WAVEFORMS (Continued)

CONTROL TIMING



2911A-1
PCM CODEC—A LAW
8-BIT COMPANDED A/D AND D/A CONVERTER

- **Per Channel, Single Chip Codec**
- **CCITT G711 and G732 Compatible, Even Order Bits Inversion Included**
- **Microcomputer Interface with On-Chip Time-Slot Computation**
- **Simple Direct Mode Interface When Fixed Timeslots Are Used**
- **±5% Power Supplies: +12V, +5V, −5V**
- **66 dB Dynamic Range, with Resolution Equivalent to 11-Bit Linear Conversion Around Zero**
- **Precision On-Chip Voltage Reference**
- **Low Power Consumption 230 mW Typ. Standby Power 33 mW Typ.**
- **Fabricated with Reliable N-Channel MOS Process**

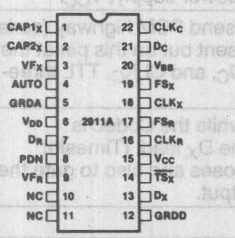
The Intel 2911A is a fully integrated PCM (Pulse Code Modulation) Codec (Coder-Decoder), fabricated with N-channel silicon gate technology. The high density of integration allows the sample and hold circuits, the digital-to-analog converter, the comparator and the successive approximation register to be integrated on the same chip, along with the logic necessary to interface a full duplex PCM link.

The primary applications are in telephone systems:

- Transmission — 30/32 Channel Systems at 2.048 Mbps
- Switching — Digital PBX's and Central Office Switching Systems
- Concentration — Subscriber Carrier/Concentrators

The wide dynamic range of the 2911A (66 dB) and the minimal conversion time (80 μs minimum) make it an ideal product for other applications, like:

- Data Acquisition
- Telemetry
- Secure Communications Systems
- Signal Processing Systems

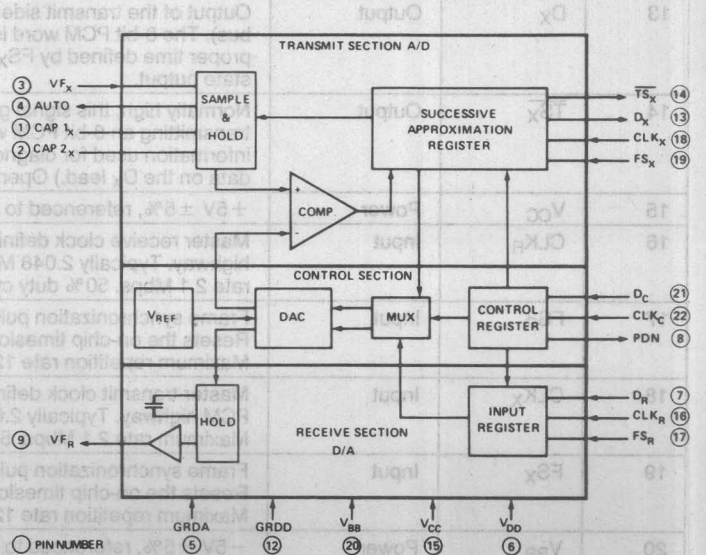


270158-1

Figure 1. Pin Configuration

CAP 1_X, CAP 2_X	Holding Capacitor
VF_X	Analog Input
VF_R	Analog Output
D_R, D_C	Digital Input
D_X, $\overline{TS_X}$	Digital Output
CLK_C, CLK_X, CLK_R	Clock Input
FS_X, FS_R	Frame Sync Input
AUTO	Auto Zero Output
V_{BB}	Power (−5V)
V_{CC}	Power (+5V)
V_{DD}	Power (+12V)
PDN	Power Down
GRDA	Analog Ground
GRDD	Digital Ground
NC	No Connect

Figure 2. Pin Names



270158-2

Figure 3. Block Diagram

November 1986
Order Number: 270158-001

PIN DESCRIPTION

Pin No	Symbol	Function	Description
1	$CAP1_X$	Hold	Connections for the transmit holding capacitor. Refer to Applications section.
2	$CAP2_X$		
3	VF_X	Input	Analog input to be encoded into a PCM word. The signal on this lead is sampled at the same rate as the transmit frame synchronization pulse FS_X, and the sample value is held in the external capacitor connected to the $CAP1_X$ and $CAP2_X$ leads until the encoding process is completed.
4	AUTO	Output	Most significant bit of the encoded PCM word (+5V for negative, −5V for positive values). Refer to the Codec Applications section.
5	GRDA	Ground	Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.
6	V_{DD}	Power	+12V ±5%; referenced to GRDA.
7	D_R	Input	Receive PCM highway (serial bus) interface. The Codec serially receives a PCM word (8 bits) through this lead at the proper time defined by FS_R, CLK_R, D_C, and CLK_C.
8	PDN	Output	Active high when the Codec is in the power down state. Open drain output.
9	VF_R	Output	Analog Output. The voltage present on VF_R is the decoded value of the PCM word received on lead D_R. This value is held constant between two conversions.
10	NC	No Connects	Recommended practice is to strap these NC's to GRDA.
11	NC		
12	GRDD	Ground	Ground return common to the logic power supply; V_{CC}.
13	D_X	Output	Output of the transmit side onto the send PCM highway (serial bus). The 8-bit PCM word is serially sent out on this pin at the proper time defined by FS_X, CLK_X, D_C, and CLK_C. TTL three-state output.
14	$\overline{TS_X}$	Output	Normally high, this signal goes low while the Codec is transmitting an 8-bit PCM word on the D_X lead. (Timeslot information used for diagnostic purposes and also to gate the data on the D_X lead.) Open drain output.
15	V_{CC}	Power	+5V ±5%, referenced to GRDD.
16	CLK_R	Input	Master receive clock defining the bit rate on the receive PCM highway. Typically 2.048 Mbps for a carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL compatible.
17	FS_R	Input	Frame synchronization pulse for the receive PCM highway. Resets the on-chip timeslot counter for the receive side. Maximum repetition rate 12 KHz. TTL interface.
18	CLK_X	Input	Master transmit clock defining the bit rate on the transmit PCM highway. Typically 2.048 Mbps for a carrier system. Maximum rate 2.1 Mbps. 50% duty cycle. TTL interface.
19	FS_X	Input	Frame synchronization pulse for the transmit PCM highway. Resets the on-chip timeslot counter for the transmit side. Maximum repetition rate 12 KHz. TTL interface.
20	V_{BB}	Power	−5V ±5%, referenced to GRDA.
21	D_C	Input	Data input to program the Codec for the chosen mode of operation. Becomes an active low chip select when CLK_C is tied to V_{CC}. TTL interface.
22	CLK_C	Input	Clock input to clock in the data on the D_C lead when the timeslot assignment feature is used; tied to V_{CC} to disable this feature. TTL interface.

FUNCTIONAL DESCRIPTION

The 2911A PCM Codec provides the analog-to-digital and the digital-to-analog conversions necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. The Codec is intended to be used on line and trunk terminations.

In a typical telephone system the Codec is located between the PCM highways and the channel filters.

The Codec encodes the incoming analog signal at the frame rate (FS_X) into an 8-bit PCM word which is sent out on the D_X lead at the proper time. Similarly, on the receive link, the Codec fetches an 8-bit PCM word from the receive highway (D_R lead) and decodes an analog value which will remain constant on lead VF_R until the next receive frame. Transmit and receive frames are independent. They can be asynchronous (transmission) or synchronous (switching) with each other.

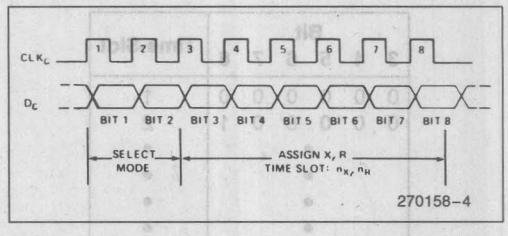
Circuitry is provided within the Codec to internally define the transmit and receive timeslots. In small systems this may eliminate the need for any external timeslot exchange; in large systems it provides one level of concentration. This feature can be bypassed and discrete timeslots sent to each Codec within a system.

In the power-down mode, most functions of the Codec are directly disabled to reduce power dissipation to a minimum.

CODEC OPERATION

Codec Control

The operation of the 2911A is defined by serially loading an 8-bit word through the D_C lead (data) and the CLK_C lead (clock). The loading is synchronous with the other operations of the Codec, and takes place whenever transitions occur on the CLK_C lead. The D_C input is loaded in during the trailing edge of the CLK_C input.



The control word contains two fields:

Bit 1 and Bit 2 define whether the subsequent 6 bits apply to both the transmit and receive side (00), the transmit side only (01), the receive side only (10), or whether the Codec should go into the standby, power-down mode (11). In the last case (11), the following 6 bits are irrelevant.

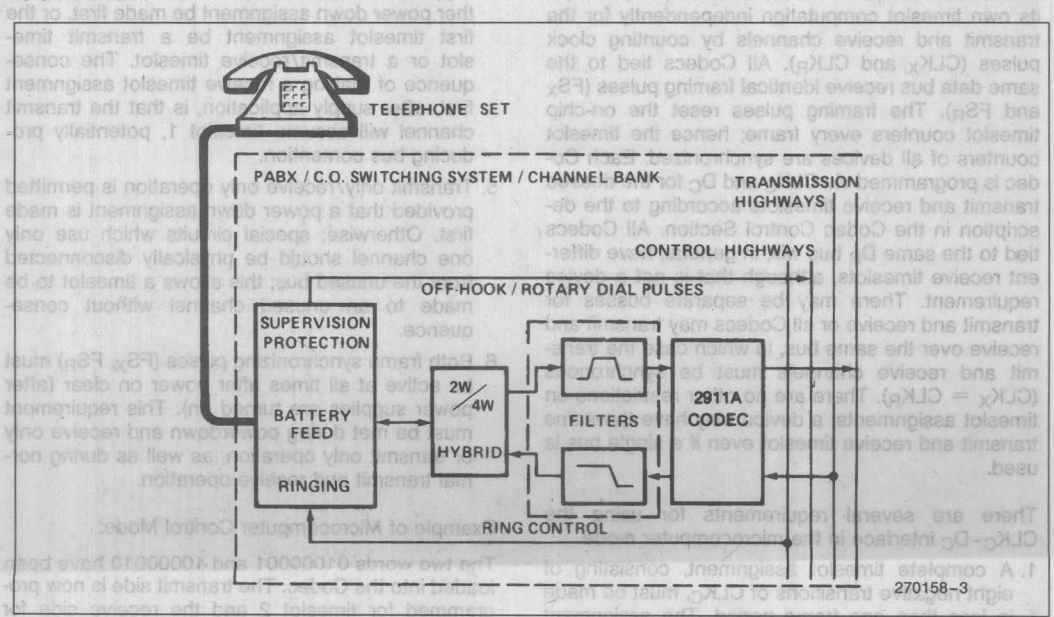


Figure 4. Typical Line Termination

The last 6 bits of the control word define the timeslot assignment, from 000000 (timeslot 1) to 111111 (timeslot 64). Bit 3 is the most significant bit and bit 8 the least significant bit and last into the Codec.

Bit 1	Bit 2	Mode
0	0	X & R
0	1	X
1	0	R
1	1	Standby

Bit 3	4	5	6	7	8	Time-Slot
0	0	0	0	0	0	1
0	0	0	0	0	1	2
			•			•
			•			•
			•			•
			•			•
1	1	1	1	1	1	64

The Codec will retain the control word (or words) until a new word is loaded in or until power is lost. This feature permits dynamic allocation of timeslots for switching applications.

Microcomputer Control Mode

In the microcomputer mode, each Codec performs its own timeslot computation independently for the transmit and receive channels by counting clock pulses (CLK_X and CLK_R). All Codecs tied to the same data bus receive identical framing pulses (FS_X and FS_R). The framing pulses reset the on-chip timeslot counters every frame; hence the timeslot counters of all devices are synchronized. Each Codec is programmed via CLK_C and D_C for the desired transmit and receive timeslots according to the description in the Codec Control Section. All Codecs tied to the same D_R bus will, in general, have different receive timeslots, although that is not a device requirement. There may be separate busses for transmit and receive or all Codecs may transmit and receive over the same bus, in which case the transmit and receive channels must be synchronous ($CLK_X = CLK_R$). There are no other restrictions on timeslot assignments; a device may have the same transmit and receive timeslot even if a single bus is used.

There are several requirements for using the CLK_C–D_C interface in the microcomputer mode.

1. A complete timeslot assignment, consisting of eight negative transitions of CLK_C, must be made in less than one frame period. The assignment can overlap a framing pulse so long as all 8 control bits are clocked in within a total span of 125 μs (for an 8 KHz frame rate). CLK_C must be left at a TTL low level when not assigning a timeslot.
2. A dead period of two frames must always be observed between successive timeslot assignments. The two frame delay is measured from the rising edge of the first CLK_C transition of the previous timeslot assigned.
3. When the device is in the power-down state (Standby), the following three-step sequence must be followed to power-up the codec to avoid contention on the transmit PCM highway.
   a. Assign a dummy transmit timeslot. The dummy should be at least two timeslots greater than the maximum valid system (usually 24 or 32). For example, in a 24 timeslot system, the dummy could be any timeslot between 26 and 64. This will power-up the transmit side, but prevent any spurious Dx or $\overline{TS}x$ outputs.
   b. Two frames later, assign the desired transmit timeslot.
   c. Two frames later assign the desired receive timeslot.
4. Initialization sequence: The device contains an on-chip power-on clear function which guarantees that with proper sequencing of the supplies (V_{CC} or V_{DD} on last), the device will initialize with no timeslot assigned to either the transmit or receive channel. After a supply failure or whenever the supplies are applied, it is recommended that either power down assignment be made first, or the first timeslot assignment be a transmit timeslot or a transmit/receive timeslot. The consequence of making a receive timeslot assignment first, after supply application, is that the transmit channel will assume timeslot 1, potentially producing bus contention.
5. Transmit only/receive only operation is permitted provided that a power down assignment is made first. Otherwise, special circuits which use only one channel should be physically disconnected from the unused bus; this allows a timeslot to be made to an unused channel without consequence.
6. Both frame synchronizing pulses (FS_X, FS_R) must be active at all times after power on clear (after power supplies are turned on). This requirement must be met during powerdown and receive only or transmit only operation, as well as during normal transmit and receive operation.

Example of Microcomputer Control Mode:

The two words 01000001 and 10000010 have been loaded into the Codec. The transmit side is now programmed for timeslot 2 and the receive side for

timeslot 3. The Codec will output a PCM word on the transmit PCM highway (bus) during the timeslot 2 of the transmit frame, and will fetch a PCM word from the receive PCM highway during timeslot 3.

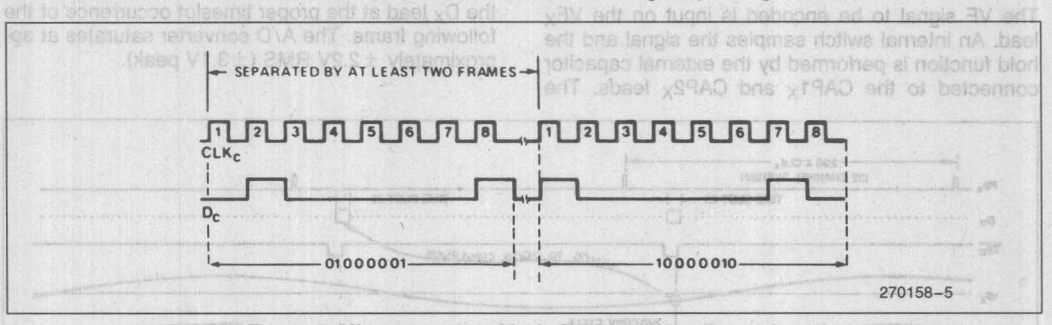


Figure 5. Microcomputer Mode Programming Examples

In this example the Codec interface to the PCM highway then functions as shown below. (FS_X and FS_R may be asynchronous.)

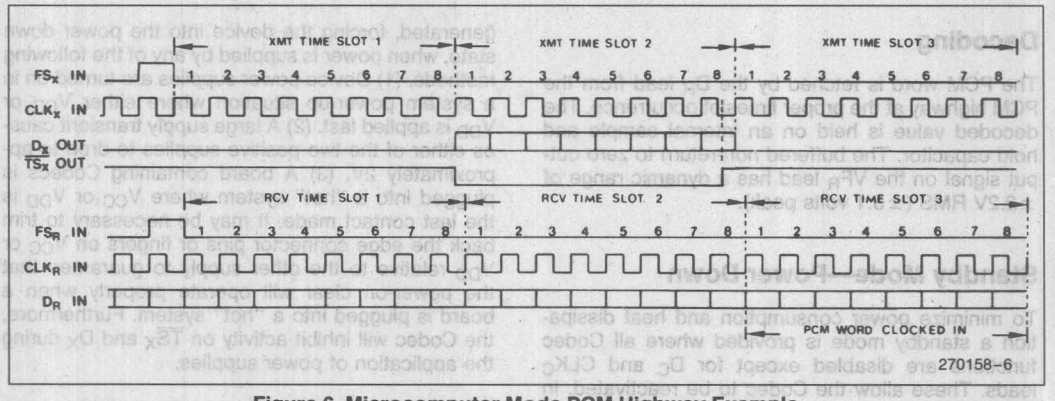


Figure 6. Microcomputer Mode PCM Highway Example

Direct Control Mode

The direct mode of operation will be selected when the CLK_C pin is strapped to the +5V supply (V_{CC}). In this mode, the D_C pin is an active low chip select. In other words, when D_C is low, the device transmits and receives in the timeslots which follow the appropriate framing pulses. With D_C high the device is in the power down state. Even though CLK_C characteristics are simpler for the 2911A it will operate properly when plugged into a 2911 board.

Deactivation of a channel by removal of the appropriate framing pulse (FS_X or FS_R) is not permitted.

Specifically, framing pulses must be applied for a minimum of two frames after a change in state of D_C in order for the D_C change to be internally sensed. In particular, when entering standby in the direct mode, framing pulses must be applied as usual for two frames after D_C is brought high.

The Codec will enter direct mode within three frame times (375 μs) as measured from the time the device power supplies settle to within the specified limits. This assumes that CLK_C is tied to V_{CC} and that all clocks are available at the time the supplies have settled.

General Control Requirements

All bit and frame clocks should be applied whenever the device is active. In particular, an unused channel cannot be deactivated by removal of its associated frame or bit clock while the other channel of the same device remains active.

A single channel cannot be deactivated except by physical disconnection of the data lead (D_X or D_R) from the system data bus. A device (both transmit and receive channels) may be deactivated in either control mode by powering down the device. Both channels are always powered down together.

Encoding

The VF signal to be encoded is input on the VF_X lead. An internal switch samples the signal and the hold function is performed by the external capacitor connected to the $CAP1_X$ and $CAP2_X$ leads. The sampling and conversion is synchronized with the transmit timeslot. The PCM word is then output on the D_X lead at the proper timeslot occurrence of the following frame. The A/D converter saturates at approximately ±2.2V RMS (±3.1V peak).

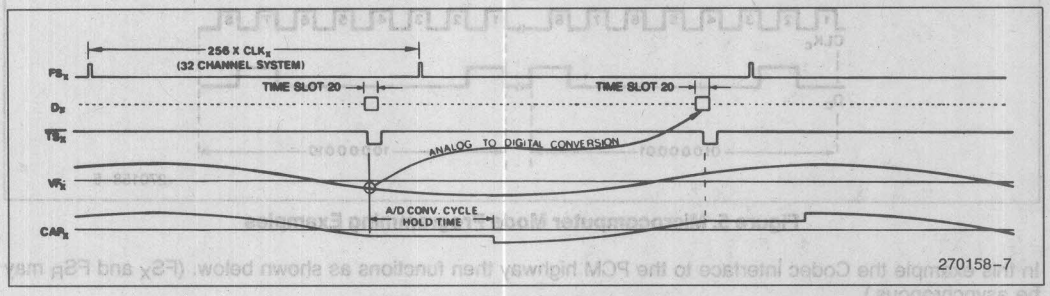


Figure 7. Transmit Encoding

Decoding

The PCM word is fetched by the D_R lead from the PCM highway at the proper timeslot occurrence. The decoded value is held on an internal sample and hold capacitor. The buffered non-return to zero output signal on the VF_R lead has a dynamic range of ±2.2V RMS (±3.1 volts peak).

Standby Mode—Power Down

To minimize power consumption and heat dissipation a standby mode is provided where all Codec functions are disabled except for D_C and CLK_C leads. These allow the Codec to be reactivated. In the microcomputer mode the Codec is placed into standby by loading a control word (D_C) with a "1" in bits 1 and 2 locations. In the direct mode when D_C is brought high, the all "1's" control word is internally transferred to the control register, invoking the standby condition.

While in the standby mode, the D_X output is actively held in a high impedance state to guarantee that the PCM bus will not be driven.

The power consumption in the standby mode is typically 33 mW.

Power-On Clear

Whether the device is used in the direct or microcomputer mode, an internal reset (power-on clear) is generated, forcing the device into the power down state, when power is supplied by any of the following methods. (1) Device power supplies are turned on in a system power-up situation where either V_{CC} or V_{DD} is applied last. (2) A large supply transient causes either of the two positive supplies to drop to approximately 2V. (3) A board containing Codecs is plugged into a "hot" system where V_{CC} or V_{DD} is the last contact made. It may be necessary to trim back the edge connector pins or fingers on V_{CC} or V_{DD} relative to the other supply to guarantee that the power-on clear will operate properly when a board is plugged into a "hot" system. Furthermore, the Codec will inhibit activity on $\overline{TS}_X$ and D_X during the application of power supplies.

The device is also tolerant of transients in the negative supply (V_{BB}) so long as V_{BB} remains more negative than −3.5V. V_{BB} transients which exceed this level should be detected and followed by a system reinitialization.

Precision Voltage Reference for the D/A Converter

The voltage reference is generated on the chip and is calibrated during the manufacturing process. The technique uses the difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature stable and bias stable reference voltage.

A gain setting op amp, programmed during manufacturing, "trims" the reference voltage source to the final precision voltage reference value provided to the D/A converter. The precision voltage reference determines the initial gain and dynamic range characteristics described in the A.C. Transmission Specification section.

CONVERSION LAW

The conversion law is commonly referred to as the A Law.

$$F(x) = Sgn(x)\left[\frac{1 + \log_{10}(A|X|)}{1 + \log_{10} A}\right], \quad 1/A \le |x| \le 1$$

$$F(x) = Sgn(x)\left[\frac{A|X|}{1 + \log_{10} A}\right], \quad 0 \le |x| \le 1/A$$

where: x = the input signal

Sgn(x) = sign of the input signal

A = 87.6 (defined by CCITT)

The Codec provides a piecewise linear approximation of the logarithmic law through 13 segments. Each segment is made of 16 steps with the exception of the first segment, which has 32 steps. In adjacent segments the step sizes are in a ratio of two to one. Within each segment, the step size is constant.

The output levels are midway between the corresponding decision levels. The output levels y_n are related to the input levels x_n by the expression:

$$y_n = \frac{x_{n-1} + x_n}{2}, \quad 0 < n \le 128$$

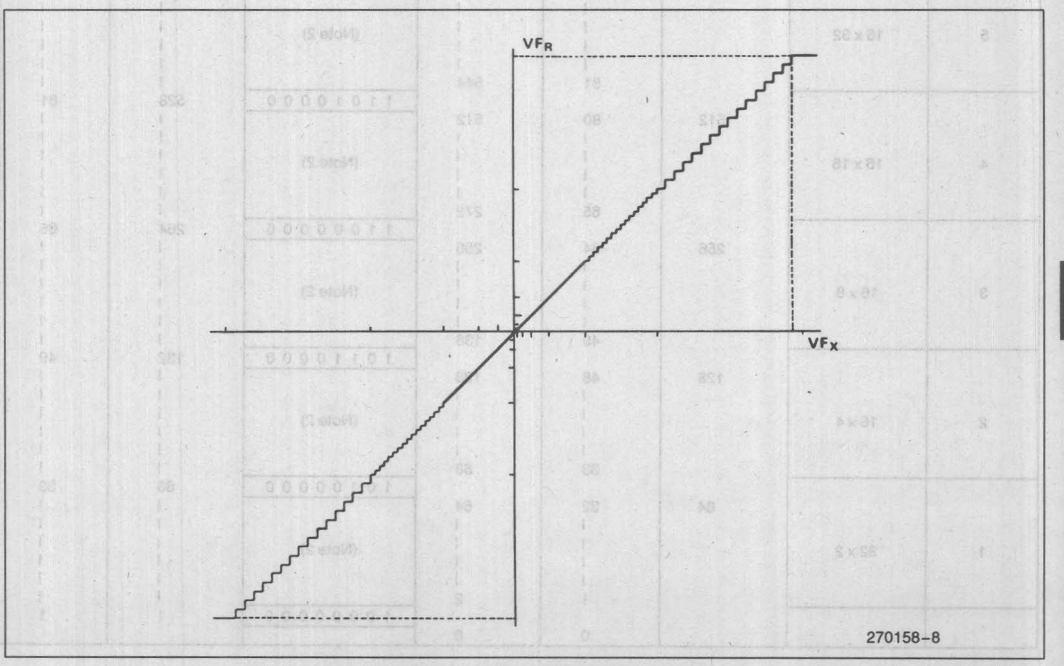


Figure 10. Codec Transfer Characteristic

5

			97	1088	1 1 1 0 0 0 0 0	1056	97
		1024	96	1024			
5	16 x 32				(Note 2)		
			81	544	1 1 0 1 0 0 0 0	528	81
		512	80	512			
4	16 x 16				(Note 2)		
			65	272	1 1 0 0 0 0 0 0	264	65
		256	64	256			
3	16 x 8				(Note 2)		
			49	136	1 0 1 1 0 0 0 0	132	49
		128	48	128			
2	16 x 4				(Note 2)		
			33	68	1 0 1 0 0 0 0 0	66	33
		64	32	64			
1	32 x 2				(Note 2)		
			1	2	1 0 0 0 0 0 0 0	1	1
			0	0			

NOTES:

1. 4096 normalized value units correspond to the value of the on-chip voltage reference.
2. The PCM word corresponding to positive input values between two successive decision values numbered n and n + 1 (see column 4) is (128 + n) expressed as a binary number.
3. X_{128} is a virtual decision value.
4. The PCM word on the highways is the same as the one shown in column 6, with the even order bits inverted. The 2911A provides for the inversion of the even order bits on both the send and receive sections.
5. The voltage output on the VF_R lead is equal to the normalized value given in the table, augmented by an offset. The offset value is approximately 15 mV.

APPLICATIONS

Holding Capacitor

For an 8 KHz sampling system the transmit holding capacitor CAP_X should be 2000 pF ±20%.

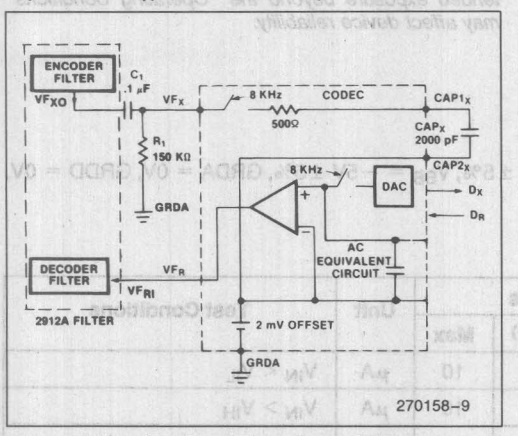


Figure 11. Circuit Interface—Without External Auto Zero

Filters Interface

The filters may be interfaced as shown in the circuit interface diagrams. Note that the output pulse stream is of the non-return-to-zero type and hence requires the (sin x)/x correction provided by the 2912A filter.

D_X Buffering

For higher drive capability or increased system reliability it may be desirable that the D_X output of a group of Codecs be buffered from the system PCM bus with an external three-state or open collector buffers. A buffer can be enabled with the appropriate Codec generated $\overline{TS}_X$ signal or signals. $\overline{TS}_X$ signal may also be used to activate external zero code suppression logic, since the occurrence of an active state of any $\overline{TS}_X$ implies the existence of PCM voice bits (as opposed to transparent data bits) on the bus.

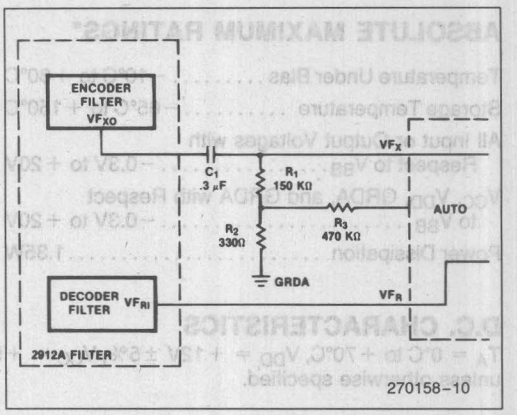


Figure 12. Circuit Interface—With External Auto Zero

Auto Zero

The 2911A contains a transparent on-chip auto zero plus a device pin for implementing a sign-bit driven external auto zero feedback loop. The on-chip auto zero reduces the input offset voltage of the encoder (VF_X) to less than 3 mV. For most telephony applications, this input offset is perfectly acceptable, since it insures the encoder is biased in the lower 25% of the first segment.

Where lower input offset is required the external auto zero loop may be used to bias the encoder exactly at the zero crossing point. The consequence of the external auto zero loop, aside from extra components, is the addition of the dithering auto-zero signal to the input signal, resulting in slightly higher idle channel noise (approximately 2 dB) than when the external loop is not used. Consequently, where the application permits, it is recommended that the external auto zero loop not be used. When not used, the AUTO pin should float.

The circuit interface with external auto zero drawing shows a possible connection between VF_X and AUTO leads with the recommended values of $C_1 = 0.3$ μF, $R_1 = 150$ KΩ, $R_2 = 330\Omega$, and $R_3 = 470$ KΩ.

5

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +80°C
Storage Temperature ..... −65°C to +150°C
All Input or Output Voltages with Respect to V_{BB} ..... −0.3V to +20V
V_{CC}, V_{DD}, GRDA, and GRDA with Respect to V_{BB} ..... −0.3V to +20V
Power Dissipation 1.35W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to +70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified.

DIGITAL INTERFACE

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
I_{IL}	Low Level Input Current			10	μA	$V_{IN} < V_{IL}$
I_{IH}	High Level Input Current			10	μA	$V_{IN} > V_{IH}$
V_{IL}	Input Low Voltage			0.6	V	
V_{IH}	Input High Voltage	2.2			V	
V_{OL}	Output Low Voltage			0.4	V	D_X, I_{OL} = 4.0 mA $\overline{TS}_X$, I_{OL} = 3.2 mA, open drain PDN, I_{OL} = 1.6 mA, open drain
V_{OH}	Output High Voltage	2.4			V	D_X, I_{OH} = 15 mA

ANALOG INTERFACE

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
Z_{AI}	Input Impedance when Sampling, VF_X	125	300	500	Ω	In series with CAP_X to GRDA, $-3.1V < V_{IN} < 3.1V$
Z_{AO}	Small Signal Output Impedance, VF_R	100	180	300	Ω	$-3.1V < V_{OUT} < 3.1V$
V_{OR}	Output Offset Voltage at VF_R	−50		50	mV	Minimum code to D_R
V_{IX}	Input Offset Voltage at VF_X	−5		5	mV	Minimum positive code produced at D_X
V_{OL}	Output Low Voltage at AUTO		V_{BB}	(V_{BB} +2)	V	400 KΩ to GRDA
V_{OH}	Output High Voltage at AUTO	(V_{CC} −2)	V_{CC}		V	400 KΩ to GRDA

D.C. CHARACTERISTICS

T_A = 0°C to +70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified. (Continued)

POWER DISSIPATION

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
I_{DDO}	Standby Current		0.7	1.1	mA	Auto Output = Open Clock Frequency = 2.048 MHZ
I_{CCO}	Standby Current		4.0	7.0	mA	
I_{BBO}	Standby Current		1.0	2.5	mA	
I_{DDI}	Operating Current		11	16	mA	
I_{CCI}	Operating Current		13	21	mA	
I_{BBI}	Operating Current		4.0	6.0	mA	

NOTE:
1. Typical values are for T_A = 25°C and nominal power supply values.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified.

TRANSMISSION

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
S/D	Signal to Total Distortion Ratio. CCITT G.712 Method 2	37			dB	Signal level 0 dBm0 to −30 dBm0
	(Half Channel)	31			dB	Signal level to −40 dBm0
		26			dB	Signal level to −45 dBm0
ΔG	2911A					VF_X = 1.02 KHz, sinusoid
	Gain Tracking Deviation		±0.25	±0.30	dB	−40 dBm0 ≤ VF_X ≤ +3 dBm0
	Half Channel(3)		±0.60	±0.70	dB	−50 dBm0 ≤ VF_X < −40 dBm0
	Reference Level −10 dBm0		±1.5	±1.8	dB	−55 dBm0 ≤ VF_X < −50 dBm0
ΔG_V	ΔG Variation with Supplies		±0.0002	±0.0004	dB/mV	−40 dBm0 ≤ VF_X ≤ +3 dBm0
	Half Channel		±0.0004	±0.0008	dB/mV	−50 dBm0 ≤ VF_X < −40 dBm0
ΔG_T	ΔG Variation with		±0.001	±0.002	dB/°C	−40 dBm0 ≤ VF_X ≤ +3 dBm0
	Temperature Half Channel		±0.002	±0.005	dB/°C	−50 dBm0 ≤ VF_X < −40 dBm0
N_{IC}	Idle Channel Noise		−85	−78	dBm0p	Quiet Code. (Note 2)
HD	Harmonic Distortion (2nd or 3rd)		−48	−44	dB	VF_X = 1.02 KHz, 0 dBm0; measured at decoder output VF_R
	Intermodulation Distortion					
IMD_1	G.712(7.1)			−45	dB	CCITT G.712
IMD_2	G.712(7.2)			−50	dBm0	Two Tone Method

5

A.C. CHARACTERISTICS

T_A = 0°C to +70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified. (Continued)

GAIN AND DYNAMIC RANGE

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
DmW	Digital Milliwatt Response	5.58	5.66	5.78	dBm	23°C, nominal supplies(4)
DmW_T	DmW_O Variation with Temperature		−0.001	−0.002	dB/°C	Relative to 23°C(4)
DmW_S	DmW_O Variation with Supplies			±0.07	dB	Supplies ±5%(4)
A_{IR}	Input Dynamic Range	2.183	2.213	2.243	V_{RMS}	Using D.C. and A.C. tests(5) 23°C, nominal supplies
A_{IRT}	Input Dynamic Range vs Temperature			−0.5	mV_{RMS}/°C	Relative to 23°C
A_{IRS}	Input Dynamic Range vs Supplies			±18	mV_{RMS}	Supplies ±5%
A_{OR}	Output Dynamic Range, VF_R	2.14	2.17	2.20	V_{RMS}	23°C, Nominal Supplies
A_{ORT}	A_{OR} Variation with Temperature			−0.5	mV_{RMS}/°C	Relative to 23°C
A_{ORS}	A_{OR} Variation with Supplies			±18	mV_{RMS}	Supplies ±5%

SUPPLY REJECTION AND CROSSTALK

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ(1)	Max		
$PSRR_1$	V_{DD} Power Supply Rejection Ratio	45			dB	decoder alone(6)
$PSRR_2$	V_{BB} Power Supply Rejection Ratio	35			dB	decoder alone(6)
$PSRR_3$	V_{CC} Power Supply Rejection Ratio	50			dB	decoder alone(6)
$PSRR_4$	V_{DD} Power Supply Rejection Ratio	50			dB	encoder alone(7)
$PSRR_5$	V_{BB} Power Supply Rejection Ratio	45			dB	encoder alone(7)
$PSRR_6$	V_{CC} Power Supply Rejection Ratio	50			dB	encoder alone(7)
CT_R	Crosstalk Isolation, Receive Side	75	80		dB	(Note 8)
CT_T	Crosstalk Isolation, Transmit Side	75	80		dB	(Note 9)
CAPX	Input Sample and Hold Capacitor	1600	2000	2400	pF	

NOTES:

1. Typical values are for T_A = 25°C and nominal power supply values.
2. If the external auto zero is used N_{IC} has a typical value of −76 dBm0.
3. Tested and guaranteed at 23°C, nominal supplies.
4. D_R of Device Under Test (D.U.T.) driven with repetitive digital word sequence specified in CCITT recommendation G.711. Measurement made at VF_R output.
5. With the D.C. method the positive and negative clipping levels are measured and A_{IR} is calculated. With the A.C. method a sinusoidal input signal to VF_X is used where A_{IR} is measured directly.
6. D.U.T. decoder; impose 200 mV_{PP}, 1.02 KHz on appropriate supply; measurement made at decoder output; decoder in idle channel conditions.
7. D.U.T. encoder, impose 200 mV_{PP}, 1.02 KHz on appropriate supply; meaurement made at encoder output; encoder in idle channel conditions.
8. VF_X of D.U.T encoder = 1.02 KHz, 0 dBm0. Decoder under quiet channel conditions; measurements made at decoder output.
9. VF_X = 0 Vrms. Decoder = 1.02 KHz, 0 dBm0. Encoder under quiet channel conditions; measurement made at encoder output.

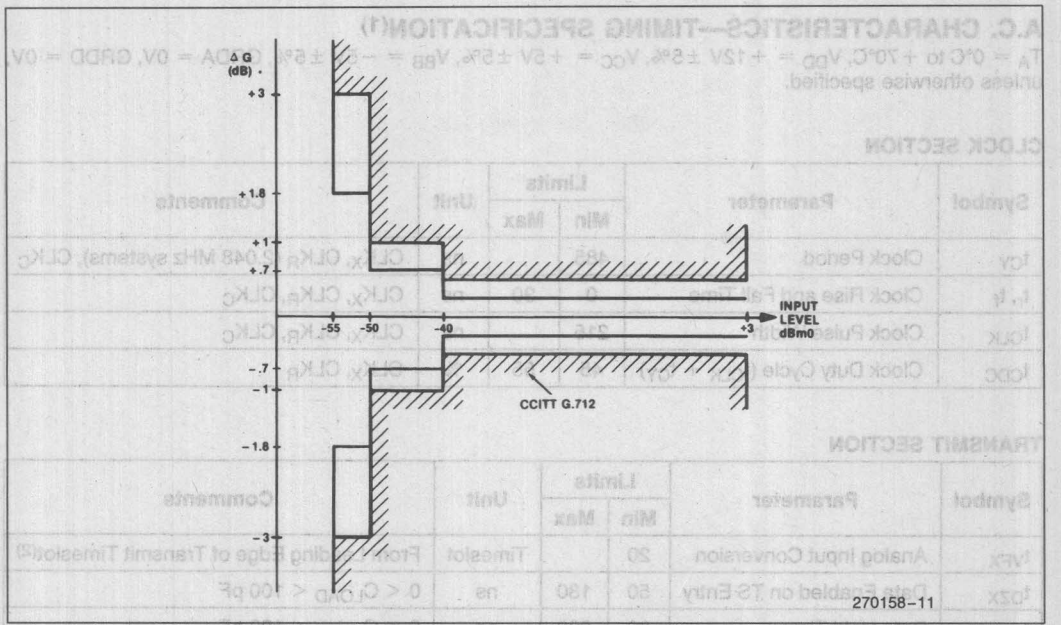


Figure 13. Tracking Deviation (ΔG) (Half Channel)

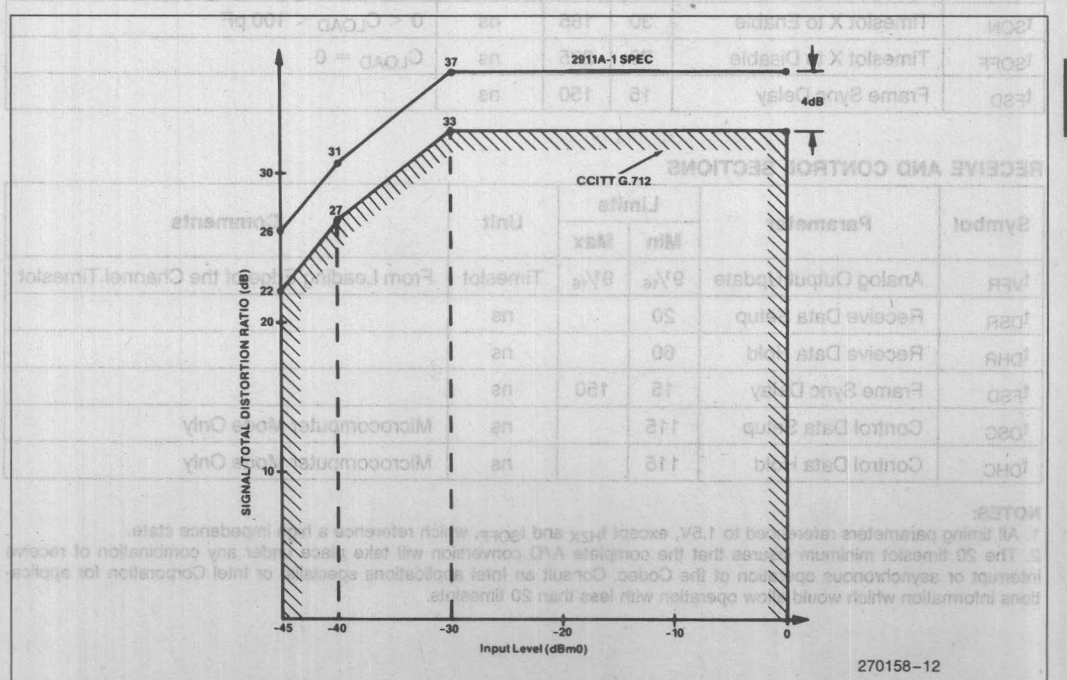


Figure 14. Signal to Total Distortion Ratio (Half Channel)

A.C. CHARACTERISTICS—TIMING SPECIFICATION(1)

T_A = 0°C to +70°C, V_{DD} = +12V ±5%, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified.

CLOCK SECTION

Symbol	Parameter	Limits		Unit	Comments
		Min	Max		
t_{CY}	Clock Period	485		ns	CLK_X, CLK_R (2.048 MHz systems), CLK_C
t_r, t_f	Clock Rise and Fall Time	0	30	ns	CLK_X, CLK_R, CLK_C
t_{CLK}	Clock Pulse Width	215		ns	CLK_X, CLK_R, CLK_C
t_{CDC}	Clock Duty Cycle ($t_{CLK} + t_{CY}$)	45	55	%	CLK_X, CLK_R

TRANSMIT SECTION

Symbol	Parameter	Limits		Unit	Comments
		Min	Max		
t_{VFX}	Analog Input Conversion	20		Timeslot	From Leading Edge of Transmit Timeslot(2)
t_{DZX}	Data Enabled on TS Entry	50	180	ns	$0 < C_{LOAD} < 100$ pF
t_{DHX}	Data Hold Time	80	230	ns	$0 < C_{LOAD} < 100$ pF
t_{HZX}	Data Float on TS Exit	75	245	ns	$C_{LOAD} = 0$
t_{SON}	Timeslot X to Enable	30	185	ns	$0 < C_{LOAD} < 100$ pF
t_{SOFF}	Timeslot X to Disable	70	225	ns	$C_{LOAD} = 0$
t_{FSD}	Frame Sync Delay	15	150	ns	

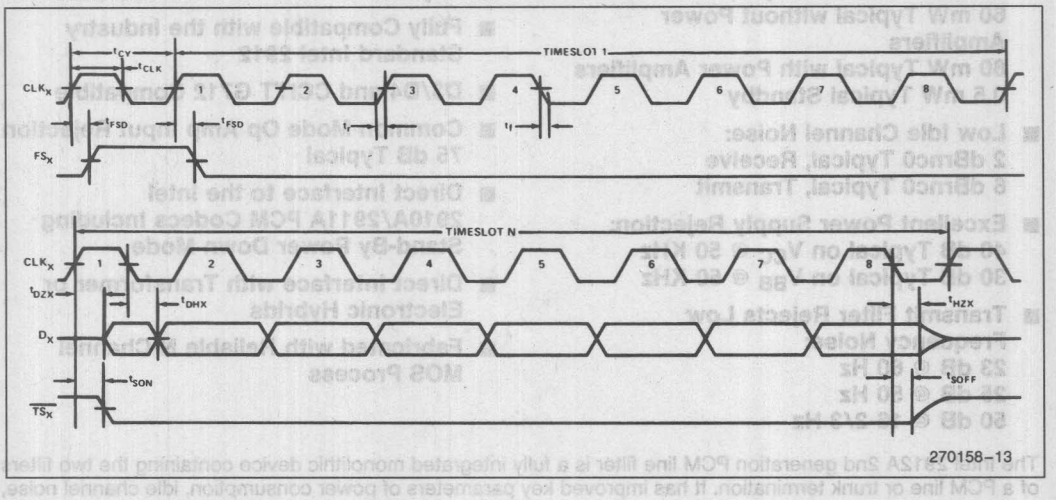
RECEIVE AND CONTROL SECTIONS

Symbol	Parameter	Limits		Unit	Comments
		Min	Max		
t_{VFR}	Analog Output Update	9 1/16	9 1/16	Timeslot	From Leading Edge of the Channel Timeslot
t_{DSR}	Receive Data Setup	20		ns	
t_{DHR}	Receive Data Hold	60		ns	
t_{FSD}	Frame Sync Delay	15	150	ns	
t_{DSC}	Control Data Setup	115		ns	Microcomputer Mode Only
t_{DHC}	Control Data Hold	115		ns	Microcomputer Mode Only

NOTES:
1. All timing parameters referenced to 1.5V, except t_{HZX} and t_{SOFF}, which reference a high impedance state.
2. The 20 timeslot minimum insures that the complete A/D conversion will take place under any combination of receive interrupt or asynchronous operation of the Codec. Consult an Intel applications specialist or Intel Corporation for applications information which would allow operation with less than 20 timeslots.

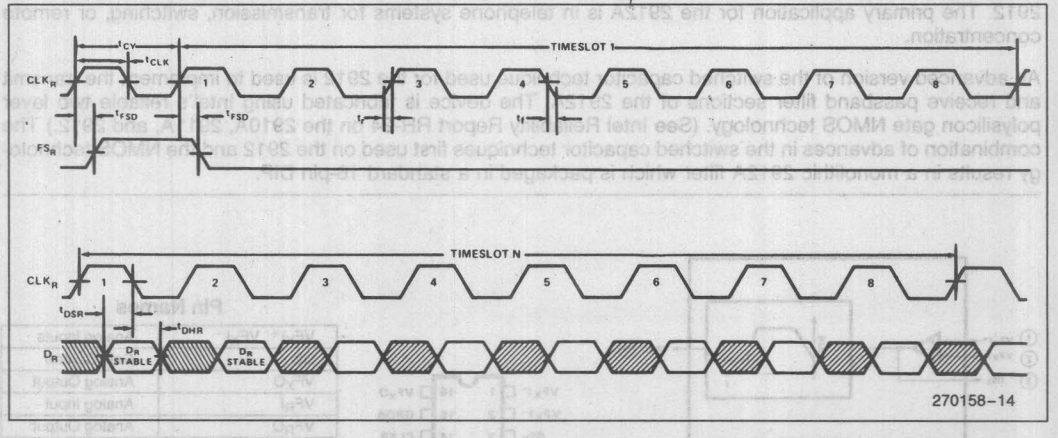
TIMING WAVEFORMS(1)

TRANSMIT TIMING



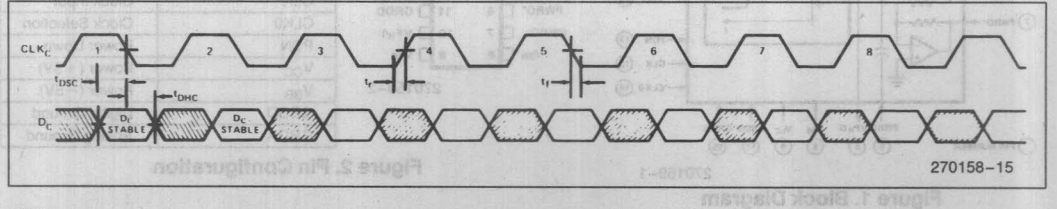
270158–13

RECEIVE TIMING



270158–14

CONTROL TIMING



270158–15

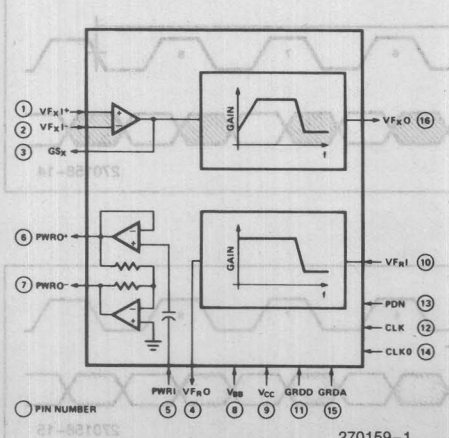
NOTE:
1. *All timing* parameters referenced to 1.5V, except t_{HZX} and t_{SOFF} which reference a high impedance state.

2912A
PCM TRANSMIT/RECEIVE FILTER

- **Low Power Consumption:**
  60 mW Typical without Power Amplifiers
  80 mW Typical with Power Amplifiers
  0.5 mW Typical Standby
- **Low Idle Channel Noise:**
  2 dBrnc0 Typical, Receive
  6 dBrnc0 Typical, Transmit
- **Excellent Power Supply Rejection:**
  40 dB Typical on V_{CC} @ 50 KHz
  30 dB Typical on V_{BB} @ 50 KHz
- **Transmit Filter Rejects Low Frequency Noise:**
  23 dB @ 60 Hz
  25 dB @ 50 Hz
  50 dB @ 16-2/3 Hz
- **Adjustable Gain in Both Directions**
- **Fully Compatible with the Industry Standard Intel 2912**
- **D3/D4 and CCITT G712 Compatible**
- **Common Mode Op Amp Input Rejection 75 dB Typical**
- **Direct Interface to the Intel 2910A/2911A PCM Codecs Including Stand-By Power Down Mode**
- **Direct Interface with Transformer or Electronic Hybrids**
- **Fabricated with Reliable N-Channel MOS Process**

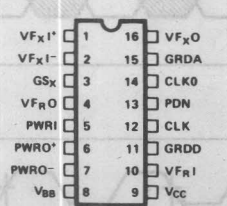
The Intel 2912A 2nd generation PCM line filter is a fully integrated monolithic device containing the two filters of a PCM line or trunk termination. It has improved key parameters of power consumption, idle channel noise, and power supply rejection. A single part exceeds both AT&T* D3/D4 and CCITT transmission specs, exceeds digital Class 5 central office switching system stringent specifications, and is fully compatible with the 2912. The primary application for the 2912A is in telephone systems for transmission, switching, or remote concentration.

An advanced version of the switched capacitor technique used for the 2912 is used to implement the transmit and receive passband filter sections of the 2912A. The device is fabricated using Intel's reliable two layer polysilicon gate NMOS technology. (See Intel Reliability Report RR-24 on the 2910A, 2911A, and 2912.) The combination of advances in the switched capacitor techniques first used on the 2912 and the NMOS technology results in a monolithic 2912A filter which is packaged in a standard 16-pin DIP.



270159-1

Figure 1. Block Diagram



270159-2

Pin Names

VF_XI^+, VF_XI^-	Analog Inputs
GS_X	Gain Control
VF_XO	Analog Output
VF_RI	Analog Input
VF_RO	Analog Output
PWRI	Driver Input
$PWRO^+$, $PWRO^-$	Driver Output
CLK	Clock Input
CLK0	Clock Selection
PDN	Power Down
V_{CC}	Power (+5V)
V_{BB}	Power (−5V)
GRDD	Digital Ground
GRDA	Analog Ground

Figure 2. Pin Configuration

*AT&T is a registered trademark of American Telephone and Telegraph Corporation.

September 1988
Order Number: 270159-002

Table 1. Pin Description

Symbol	Pin No.	Function	Description
VF_XI+	1	Input	Analog input of the transmit filter. The VF_XI+ signal comes from the 2 to 4 wire hybrid in the case of a 2 wire line and goes through the frequency rejection and the antialiasing filters before being sent to the Codec for encoding.
VF_XI-	2	Input	Inverting input of the gain adjustment operational amplifier on the transmit filter.
GS_X	3	Output	Output of the gain adjustment operational amplifier on the transmit filter. Used for gain setting of the transmit filter.
VF_RO	4	Output	Analog output of the receive filter. This output provides a direct interface to electronic hybrids. For a transformer hybrid application, VF_RO is tied to PRWI and a dual balanced output is provided on pins PWRO+ and PWRO−.
PWRI	5	Input	Input to the power driver amplifiers on the receive side for interface to transformer hybrids. High impedance input. When tied to V_{BB}, the power amplifiers are powered down.
PWRO+	6	Output	Non-inverting side of the power amplifiers. Power driver output capable of directly driving transformer hybrids.
PWRO−	7	Output	Inverting side of the power amplifiers. Power driver output capable of directly driving transformer hybrids.
V_{BB}	8	Power	−5V ±5% referenced to GRDA
V_{CC}	9	Power	+5V ±5% referenced to GRDA
VF_RI	10	Input	Analog input of the receive filter, interface to the Codec analog output for PCM applications. The receive filter provides the $\frac{\text{Sin}x}{x}$ correction needed for sample and hold type Codec outputs to give unity gain. The input voltage range is directly compatible with the Intel 2910A and 2911A Codecs.
GRDD	11	Ground	Digital ground return for internal clock generator.
CLK(1)	12	Input	Clock input. Three clock frequencies can be used: 1.536 MHz, 1.544 MHz or 2.048 MHz; pin 14, CLK0, has to be strapped accordingly. High impedance input, TTL voltage levels.
PDN	13	Input	Control input for the stand-by power down mode. An internal pull up to +5V is provided for interface to the Intel 2910A and 2911A PDN outputs. TTL voltage levels.
CLK0(1)	14	Input	Clock (pin 12, CLK) frequency selection. If tied to V_{BB}, CLK should be 1.536 MHz. If tied to Ground, CLK should be 1.544 MHz. If tied to V_{CC}, CLK should be 2.048 MHz.
GRDA	15	Ground	Analog return common to the transmit and receive analog circuits. Not connected to GRDD internally.
VF_XO	16	Output	Analog output of the transmit filter. The output voltage range is directly compatible with the Intel 2910A and 2911A Codecs.

NOTE:
1. The three clock frequencies are directly compatible with the Intel 2910A and 2911A Codecs. The following table should be observed in selecting the clock frequency.

Codec Clock	Clock Bits/Frame	CLK, Pin 12	CLK0, Pin 14
1.536 MHz	192	1.536 MHz	V_{BB} (−5V)
1.544 MHz	193	1.544 MHz	GRDD
2.048 MHz	256	2.048 MHz	V_{CC} (+5V)

5

FUNCTIONAL DESCRIPTION

The 2912A provides the transmit and receive filters found on the analog termination of a PCM line or trunk. The transmit filter performs the anti-aliasing function needed for an 8 KHz sampling system, and the 50/60 Hz rejection. The receive filter has a low pass transfer characteristic and also provides the Sinx/x correction necessary to interface the Intel 2910A (μ Law) and 2911A (A Law) Codecs which have a non-return-to-zero output of the digital to analog conversion. Gain adjustment is provided in the receive and transmit directions.

A stand-by, power down mode is included in the 2912A and can be directly controlled by the 2910A/2911A Codecs.

The 2912A can interface directly with a transformer hybrid (2 to 4 wire conversion) or with electronic hybrids; in the latter case the power dissipation is reduced by powering down the output amplifier provided on the 2912A.

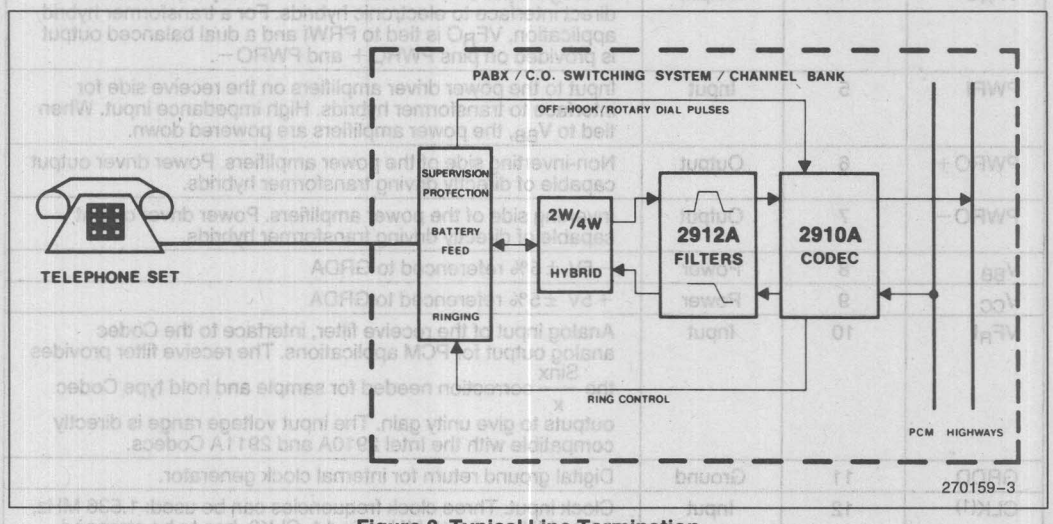


Figure 3. Typical Line Termination

FILTER OPERATION

Transmit Filter Input Stage

The input stage provides gain adjustment in the pass-band. The input operational amplifier has a common mode range of ± 2.2 volts, a DC offset of less than 25 mV, a voltage gain greater than 3000 and a unity gain bandwidth of 1 MHz. It can be connected to provide a gain of 20 dB without degrading the noise performance of the filter. The load impedance connected to the amplifier output (GS_X) must be greater than 10K Ω in parallel with 25 pF. The input signal on lead VF_XI+ can be either AC or DC coupled. The input Op Amp can also be used in the inverting mode or differential amplifier mode. The remaining portion of the transmit filter provides a gain of +3 dB in the pass band.

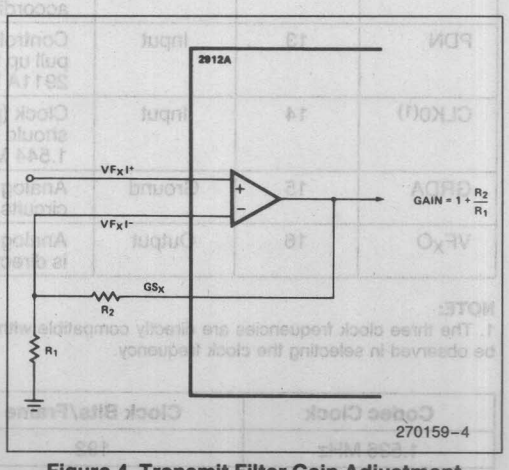


Figure 4. Transmit Filter Gain Adjustment

Receive Filter Output

The VF_RO lead is capable of driving high impedance electronic hybrids. The gain of the receive section from VF_RI to VF_RO is:

$$\frac{\left(\frac{\pi f}{8000}\right)}{\text{Sin}\left(\frac{\pi f}{8000}\right)}$$

which when multiplied by the output response of the Intel 2910A and 2911A Codecs results in a 0 dB gain in the pass band. The filter gain can be adjusted downward by a resistor voltage divider connected as shown in Figure 5. The total resistive load R_{LR} on VF_RO should not be less than 10K Ω.

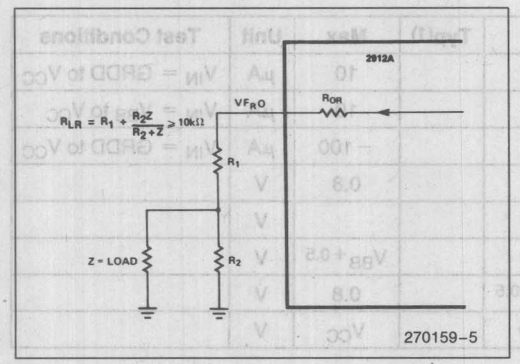


Figure 5. Receive Filter Output Gain Adjustment

Receive Filter Output Driver Amplifier Stage

A balanced power amplifier is provided in order to drive low-impedance loads in a bridged configuration. The receive filter output VF_RO is connected through gain setting resistors R_1 and R_2 to the amplifier input PWRI. The input voltage range on PWRI is ±3.2 volts and the gain is 6 dB for a bridged output.

With a 600Ω load connected between PWRO+ and PWRO−, the maximum voltage swing across the load is ±5.0 volts. The series combination of R_S and the hybrid transformer must present a minimum A.C. load resistance of 600Ω to the amplifier in the bridged configuration. A typical connection of the output driver amplifiers is shown in Figure 6. These amplifiers can also be used with loads connected to ground.

When the power amplifier is not needed it should be deactivated to save power. This is accomplished by tying the PWRI pin to V_{BB} before the device is powered up.

Power Down Mode

Pin 13, PDN, provides the power down control. When the signal on this lead is brought high, the 2912A goes into a standby, power down mode. Power dissipation is reduced to 0.5 mW. In the stand-by mode, all outputs go into a high impedance state. This feature allows multiple 2912As to drive the same analog bus on a time-shared basis.

When power is restored, the settling time of the 2912A is typically 15 ms.

The PDN interface is directly compatible with the Intel 2910A and 2911A PDN outputs. Only one command from the common control is then necessary to power down both the Codec and the Filters of the line or trunk interface.

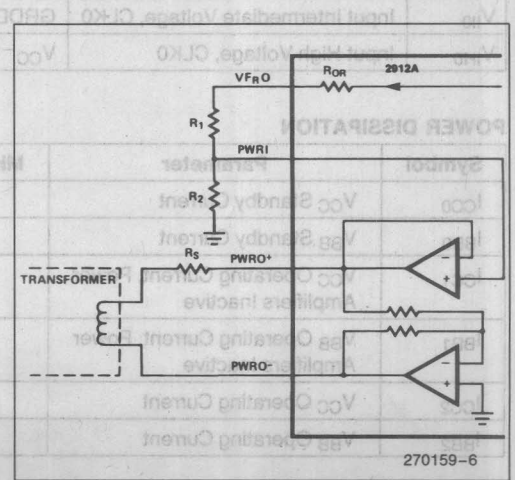


Figure 6. Typical Connection of Output Driver Amplifier

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +80°C
Storage Temperature ..... −65°C to +150°C
Supply Voltage with Respect to V_{BB} ..... −0.3V to +14.0V
All Input and Output Voltages with Respect to V_{BB} ..... −0.3V to +14.0V
All Output Currents ..... ±50 mA
Power Dissipation ..... 1 Watt

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5V ±5%; V_{BB} = −5V ±5%; GRDA = 0V; GRDD = 0V; unless otherwise specified

DIGITAL INTERFACE (CLK, CLK0, and PDN Pins)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
I_{LIC}	Input Load Current, CLK			10	μA	V_{IN} = GRDD to V_{CC}
I_{LIO}	Input Load Current, CLK0			10	μA	V_{IN} = V_{BB} to V_{CC}
I_{LIP}	Input Load Current, PDN			−100	μA	V_{IN} = GRDD to V_{CC}
V_{IL}	Input Low Voltage (except CLK0)			0.8	V	
V_{IH}	Input High Voltage (except CLK0)	2.0			V	
V_{IL0}	Input Low Voltage, CLK0	V_{BB}		V_{BB} + 0.5	V	
V_{II0}	Input Intermediate Voltage, CLK0	GRDD − 0.5		0.8	V	
V_{IH0}	Input High Voltage, CLK0	V_{CC} − 0.5		V_{CC}	V	

POWER DISSIPATION

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
I_{CC0}	V_{CC} Standby Current		50	100	μA	PDN = V_{IH} Min
I_{BB0}	V_{BB} Standby Current		50	100	μA	PDN = V_{IH} Min
I_{CC1}	V_{CC} Operating Current, Power Amplifiers Inactive		6	10	mA	PWRI = V_{BB}(2)
I_{BB1}	V_{BB} Operating Current, Power Amplifiers Inactive		6	10	mA	PWRI = V_{BB}(2)
I_{CC2}	V_{CC} Operating Current		8	14	mA	
I_{BB2}	V_{BB} Operating Current		8	14	mA	

NOTES:
1. Typical values are for T_A = 25°C and nominal power supply values.
2. To place the power amplifiers in the inactive mode PWRI must be tied to V_{BB} prior to power-up.

D.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified (Continued)

ANALOG INTERFACE, TRANSMIT FILTER INPUT STAGE

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
I_{BXI}	Input Leakage Current, VF_XI+, VF_XI-			100	nA	$-2.2V < V_{IN} < 2.2V$
R_{IXI}	Input Resistance, VF_XI+, VF_XI-	10			MΩ	
V_{OSXI}	Input Offset Voltage, VF_XI+, VF_XI-			25	mV	
CMRR	Common Mode Rejection, VF_XI+, VF_XI-	60	75		dB	$-2.2V < V_{IN} < 2.2V$, 0 dBm0 ≡ 1.1 V_{RMS}, Input at VF_XI-
A_{VOL}	DC Open Loop Voltage Gain, GS_X	3000				
f_C	Open Loop Unity Gain Bandwidth, GS_X		1		MHz	
V_{OXI}	Output Voltage Swing, GS_X	±2.5			V	$R_L \geq 10$ KΩ
C_{LXI}	Load Capacitance, GS_X			25	pF	
R_{LXI}	Minimum Load Resistance, GS_X	10			KΩ	Minimum R_L

ANALOG INTERFACE, TRANSMIT FILTER (See Figure 9)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
R_{OX}	Output Resistance, VF_XO		20	35	Ω	
V_{OSX}	Output DC Offset, VF_XO			100	mV	VF_XI+ Connected to GRDA, Input Op Amp at Unity Gain
$PSRR_1$	Power Supply Rejection of V_{CC} at 1 KHz, VF_XO	30	40		dB	Note 2
$PSRR_2$	Power Supply Rejection of V_{BB} at 1 KHz, VF_XO	25	30		dB	Note 2
C_{LX}	Load Capacitance, VF_XO			25	pF	
R_{LX}	Minimum Load Resistance, VF_XO	2.7			KΩ	Minimum R_L
V_{OX1}	Output Voltage Swing, 1 KHz, VF_XO	±3.2			V	$R_L \geq 10$ KΩ or with 2910A or 2911A
V_{OX2}	Output Voltage Swing, 1 KHz, VF_XO	±2.5			V	$R_L \geq 2.7$ KΩ

NOTES:
1. Typical values for $T_A = 25°C$ and nominal power supply values.
2. $PSRR_{1,2}$ include op amp in transmit section.

5

D.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified (Continued)

ANALOG INTERFACE, RECEIVE FILTER (See Figure 10)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
I_{BR}	Input Leakage Current, VF_RI			3	μA	$-3.2V < V_{IN} < 3.2V$
R_{IR}	Input Resistance, VF_RI	1			MΩ	
R_{OR}	Output Resistance, VF_RO			100	Ω	
V_{OSR}	Output DC Offset VF_RO			100	mV	VF_RI Connected to GRDA
$PSRR_3$	Power Supply Rejection of V_{CC} at 1 KHz, VF_RO	30	45		dB	
$PSRR_4$	Power Supply Rejection of V_{BB} at 1 KHz, VF_RO	30	35		dB	
C_{LR}	Load Capacitance, VF_RO			25	pF	
R_{LR}	Minimum Load Resistance, VF_RO	10			KΩ	Minimum R_L
V_{OR}	Output Voltage Swing, VF_RO	±3.2			V	R_L = 10 KΩ

ANALOG INTERFACE, RECEIVE FILTER DRIVER AMPLIFIER STAGE

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions	
I_{BRA}	Input Leakage Current, PWRI			3	μA	$-3.2V < V_{IN} < 3.2V$	
R_{IRA}	Input Resistance, PWRI	10			MΩ		
R_{ORA}	Output Resistance, PWRO+, PWRO−		1		Ω	$\lvert I_{OUT}\rvert < 10$ mA $-3.0V < V_{OUT} < 3.0V$	
V_{OSRA}	Output DC Offset, PWRO+, PWRO−			50	mV	PWRI Connected to GRDA	
C_{LRA}	Load Capacitance, PWRO+, PWRO−			100	pF		
V_{ORA1}	Output Voltage Swing Across R_L, PWRO+, PWRO− Single Ended Connection	±3.2			V	R_L = 10 KΩ	R_L Connected to GRDA fo ≥ 200 Hz
		±2.9			V	R_L = 600Ω	
		±2.5			V	R_L = 300Ω	
V_{ORA2}	Differential Output Voltage Swing, PWRO+, PWRO− Balanced Output Connection	±6.4			V	R_L = 20 KΩ	R_L Connected between PWRO + and PWRO − fo ≥ 200 Hz
		±5.8			V	R_L = 1200Ω	
		±5.0			V	R_L = 600Ω	

NOTE:
1. Typical values are for $T_A = 25°C$ and nominal power supply values.

A.C. CHARACTERISTICS

$T_A = 0°C$ to $+70°C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified

Clock Input Frequency: CLK = 1.536 MHz ±0.1%; CLK0 = V_{IL0} (Tied to V_{BB})
CLK = 2.048 MHz ±0.1%; CLK0 = V_{IH0} (Tied to V_{CC})
CLK = 1.544 MHz ±0.1%; CLK0 = V_{II0} (Tied to GRDD)

TRANSMIT FILTER TRANSFER CHARACTERISTICS

(See Transmit Filter Transmit Characteristics, Figure 7)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
G_{RX}	Gain Relative to Gain at 1 KHz					0 dBm0 Input Signal
	16.67 Hz		−56	−50	dB	Gain Setting Op Amp
	50 Hz			−25	dB	Unity Gain
	60 Hz			−23	dB	
	200 Hz	−1.8		−0.125	dB	0 dBm0 Signal ≡ 1.1 V_{RMS}
	300 Hz to 3000 Hz	−0.125		0.125	dB	Input at VF_XI-
	3300 Hz	−0.35		0.03	dB	
	3400 Hz	−0.7		−0.1	dB	0 dBm0 Signal ≡ 1.6 V_{RMS}
	4000 Hz			−14	dB	Output at VF_XO
	4600 Hz and Above			−32	dB	
G_{AX}	Absolute Passband Gain at 1 KHz, VF_XO	2.9	3.0	3.1	dB	$R_L = \infty$ (3)
G_{AXT}	Gain Variation with Temperature at 1 KHz		0.0002	0.002	dB/°C	0 dBm0 Signal Level
G_{AXS}	Gain Variation with Supplies at 1 KHz		0.01	0.07	dB/V	0 dBm0 Signal Level, Supplies ±5%
CT_{RT}	Cross Talk, Receive to Transmit, Measured at VF_XO $20 \log \frac{VF_XO}{VF_RO}$		−75	−65	dB	VF_RI = 1.6 V_{RMS}, 1 KHz Input, VF_XI+, VF_XI- Connected to GS_X, GS_X Connected through 10 KΩ to GRDA
N_{CX1}	Total C Message Noise at Output, VF_XO		6	11	dBrnc0 (Note 2)	Gain Setting Op Amp at Unity Gain
N_{CX2}	Total C Message Noise at Output, VF_XO		9	13	dBrnc0 (Note 2)	Gain Setting Op Amp at 20 dB Gain
D_{DX}	Differential Envelope Delay, VF_XO 1 KHz to 2.6 KHz			60	μs	
D_{AX}	Absolute Delay at 1 KHz, VF_XO			110	μs	
DP_{X1}	Single Frequency Distortion Products			−48	dB	0 dBm0 Input Signal at 1 KHz
DP_{X2}	Single Frequency Distortion Products at Maximum Signal Level of +3 dBm0 at VF_XO			−45	dB	0.16 V_{RMS} 1 KHz Input Signal at VF_XI+; Gain Setting Op Amp at 20 dB Gain. The +3 dBm0 Signal at VF_XO is 2.26 V_{RMS}

5

A.C. CHARACTERISTICS $T_A = 0°C$ to $+70°C$; $V_{CC} = 5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified (Continued)

Clock Input Frequency: CLK = 1.536 MHz ±0.1%; CLK0 = V_{IL0} (Tied to V_{BB})
CLK = 1.544 MHz ±0.1%; CLK0 = V_{II0} (Tied to GRDD)
CLK = 2.048 MHz ±0.1%; CLK0 = V_{IH0} (Tied to V_{CC})

RECEIVE FILTER TRANSFER CHARACTERISTICS (See Receive Filter Transfer Characteristics, Figure 8)

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
G_{RR}	Gain Relative to Gain at 1 KHz with Sinx/x Correction of 2910A or 2911A					0 dBm0 Input Signal
	Below 200 Hz			0.125	dB	0 dBm0 Signal ≡ 1.6 V_{RMS} × $\frac{\text{Sin}\left(\frac{\pi f}{8000}\right)}{\left(\frac{\pi f}{8000}\right)}$ Input at VF_RI
	200 Hz	−0.5		0.125	dB	
	300 Hz to 3000 Hz	−0.125		0.125	dB	
	3300 Hz	−0.35		0.03	dB	
	3400 Hz	−0.7		−0.1	dB	
	4000 Hz			−14	dB	
	4600 Hz and Above			−30	dB	
G_{AR}	Absolute Passband Gain at 1 KHz, VF_RO	−0.1	0	+0.1	dB	$R_L = \infty$ (3, 4)
G_{ART}	Gain Variation with Temperature at 1 KHz		0.0002	0.002	dB/°C	0 dBm0 Signal Level
G_{ARS}	Gain Variation with Supplies at 1 KHz		0.01	0.07	dB/V	0 dBm0 Signal Level, Supplies ±5%
CT_{TR}	Cross Talk, Transmit to Receive, Measured at VF_RO; 20 log (VF_RO/VF_XO)		−70	−60	dB	VF_XI = 1.1 V_{RMS}, 1 KHz Output, VF_RI Connected to GRDA
N_{CR}	Total C Message Noise at Output, VF_RO		2	6	dBrnc0 (Note 2)	VF_RO Output or PWRO+ and PWRO− Connected with Unity Gain
D_{DR}	Differential Envelope Delay, VF_RO, 1 KHz to 2.6 KHz			100	μs	
D_{AR}	Absolute Delay at 1 KHz, VF_RO			110	μs	
DP_{R1}	Single Frequency Distortion Products			−48	dB	0 dBm0 Input Signal at 1 KHz
DP_{R2}	Single Frequency Distortion Products at Maximum Signal Level of +3 dBm0 at VF_RO			−45	dB	+3 dBm0 Signal Level of 2.26 V_{RMS}, 1 KHz Input at VF_RI

NOTES:
1. Typical Values are for $T_A = 25°C$ and nominal power supply values.
2. A noise measurement of 12 dBrnc into a 600Ω load at the 2912A device is equivalent to 6 dBrnc0.
3. For gain under load refer to output resistance specs and perform gain calculation.
4. Output is non-inverting.

TRANSFER CHARACTERISTICS

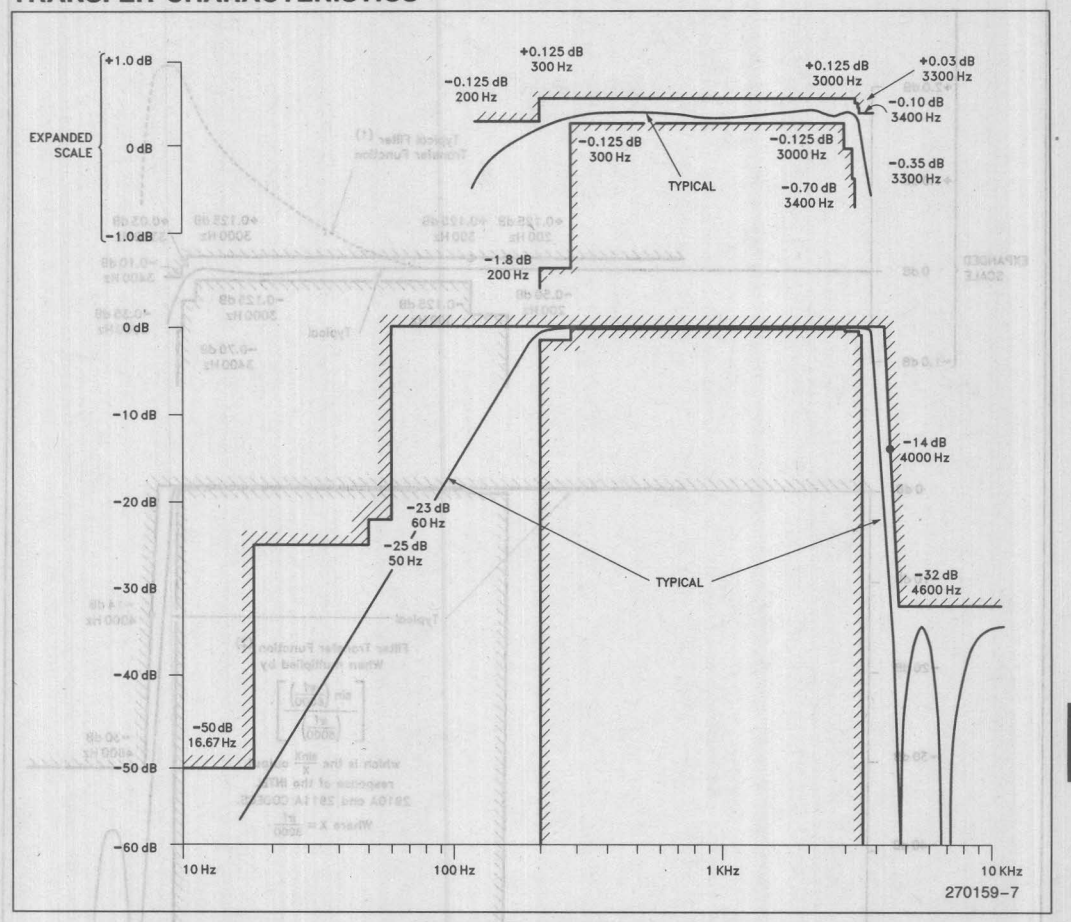


Figure 7. Transmit Filter

5

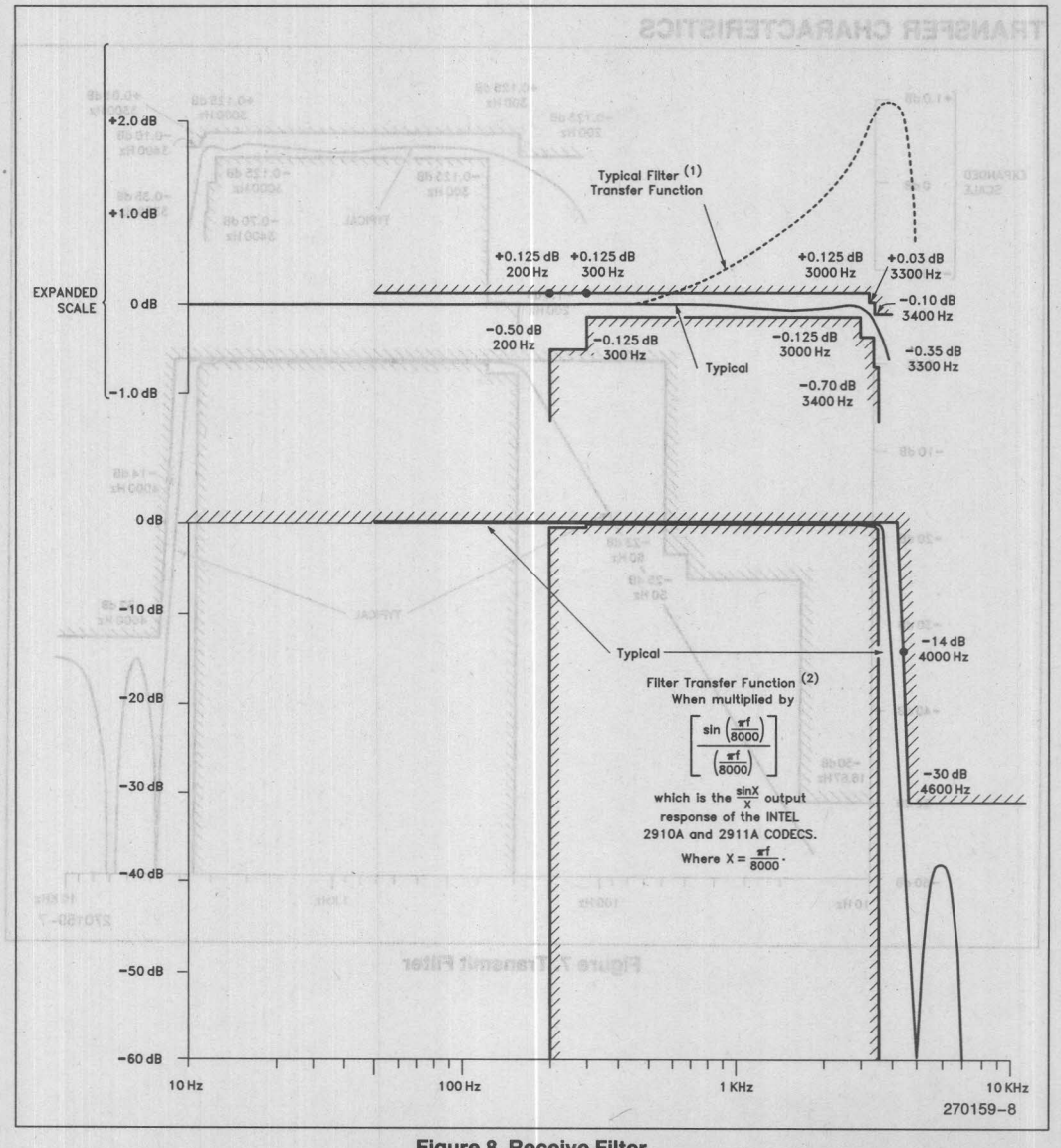


Figure 8. Receive Filter

NOTES:
1. Typical Transfer Function of the Receive Filter as a Separate Component.
2. Typical Transfer Function of the Receive Filter Driven by the Sample and Hold Output of the Intel 2910A and 2911A CODECS. The Combined Filter/CODEC Response Meets the Stated Specifications.

POWER SUPPLY REJECTION TYPICAL VALUES OVER 3 RANGES

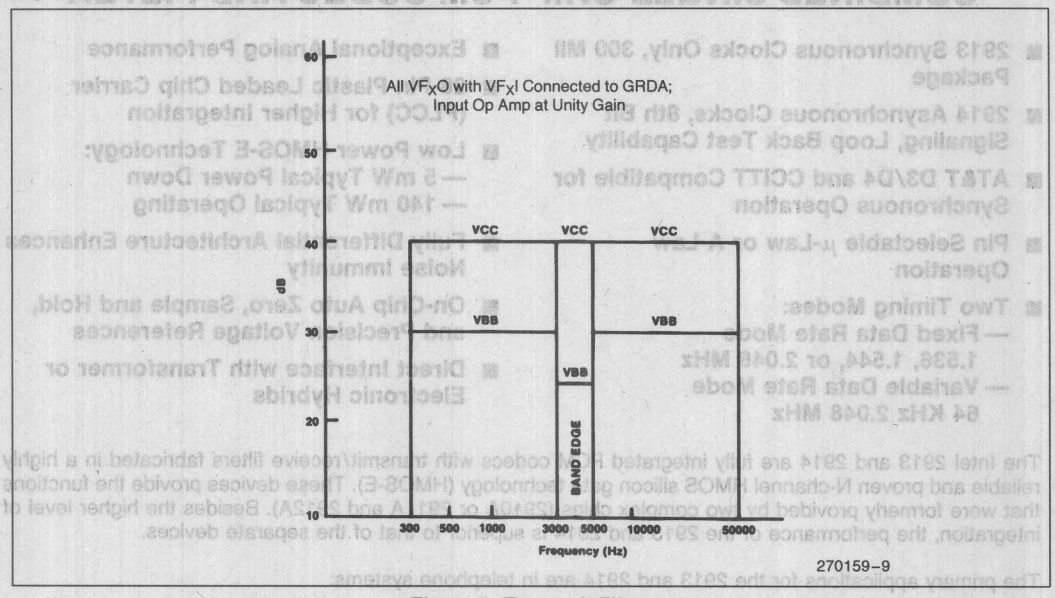


Figure 9. Transmit Filter

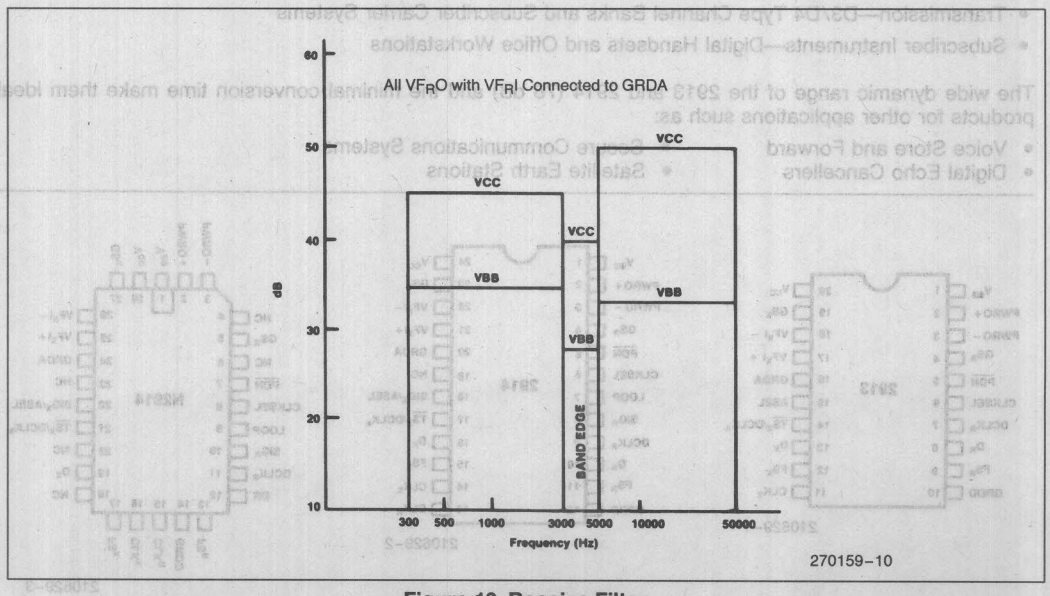


Figure 10. Receive Filter

5

2913 AND 2914
COMBINED SINGLE-CHIP PCM CODEC AND FILTER

- **2913 Synchronous Clocks Only, 300 Mil Package**
- **2914 Asynchronous Clocks, 8th Bit Signaling, Loop Back Test Capability**
- **AT&T D3/D4 and CCITT Compatible for Synchronous Operation**
- **Pin Selectable μ-Law or A-Law Operation**
- **Two Timing Modes:**
  - **— Fixed Data Rate Mode 1.536, 1.544, or 2.048 MHz**
  - **— Variable Data Rate Mode 64 KHz 2.048 MHz**
- **Exceptional Analog Performance**
- **28-Pin Plastic Leaded Chip Carrier (PLCC) for Higher Integration**
- **Low Power HMOS-E Technology:**
  - **— 5 mW Typical Power Down**
  - **— 140 mW Typical Operating**
- **Fully Differential Architecture Enhances Noise Immunity**
- **On-Chip Auto Zero, Sample and Hold, and Precision Voltage References**
- **Direct Interface with Transformer or Electronic Hybrids**

The Intel 2913 and 2914 are fully integrated PCM codecs with transmit/receive filters fabricated in a highly reliable and proven N-channel HMOS silicon gate technology (HMOS-E). These devices provide the functions that were formerly provided by two complex chips (2910A or 2911A and 2912A). Besides the higher level of integration, the performance of the 2913 and 2914 is superior to that of the separate devices.

The primary applications for the 2913 and 2914 are in telephone systems:

- Switching—Digital PBX's and Central Office Switching Systems
- Transmission—D3/D4 Type Channel Banks and Subscriber Carrier Systems
- Subscriber Instruments—Digital Handsets and Office Workstations

The wide dynamic range of the 2913 and 2914 (78 dB) and the minimal conversion time make them ideal products for other applications such as:

- Voice Store and Forward
- Digital Echo Cancellers
- Secure Communications Systems
- Satellite Earth Stations

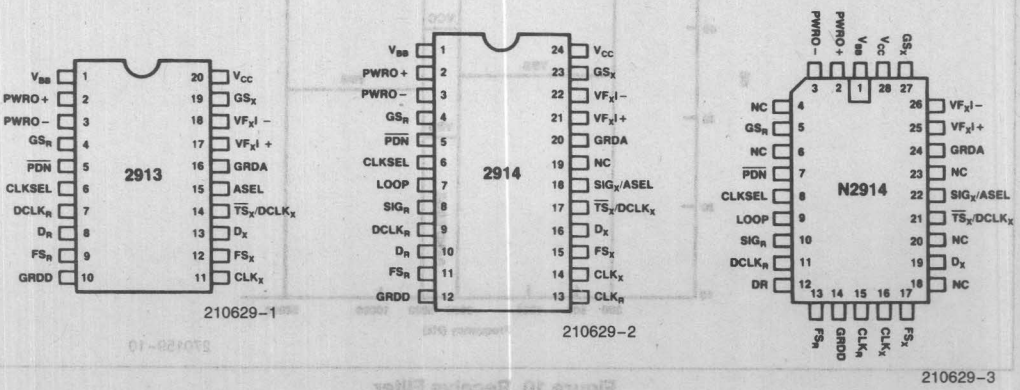


Figure 1. Pin Configurations

September 1990
Order Number: 210629-003

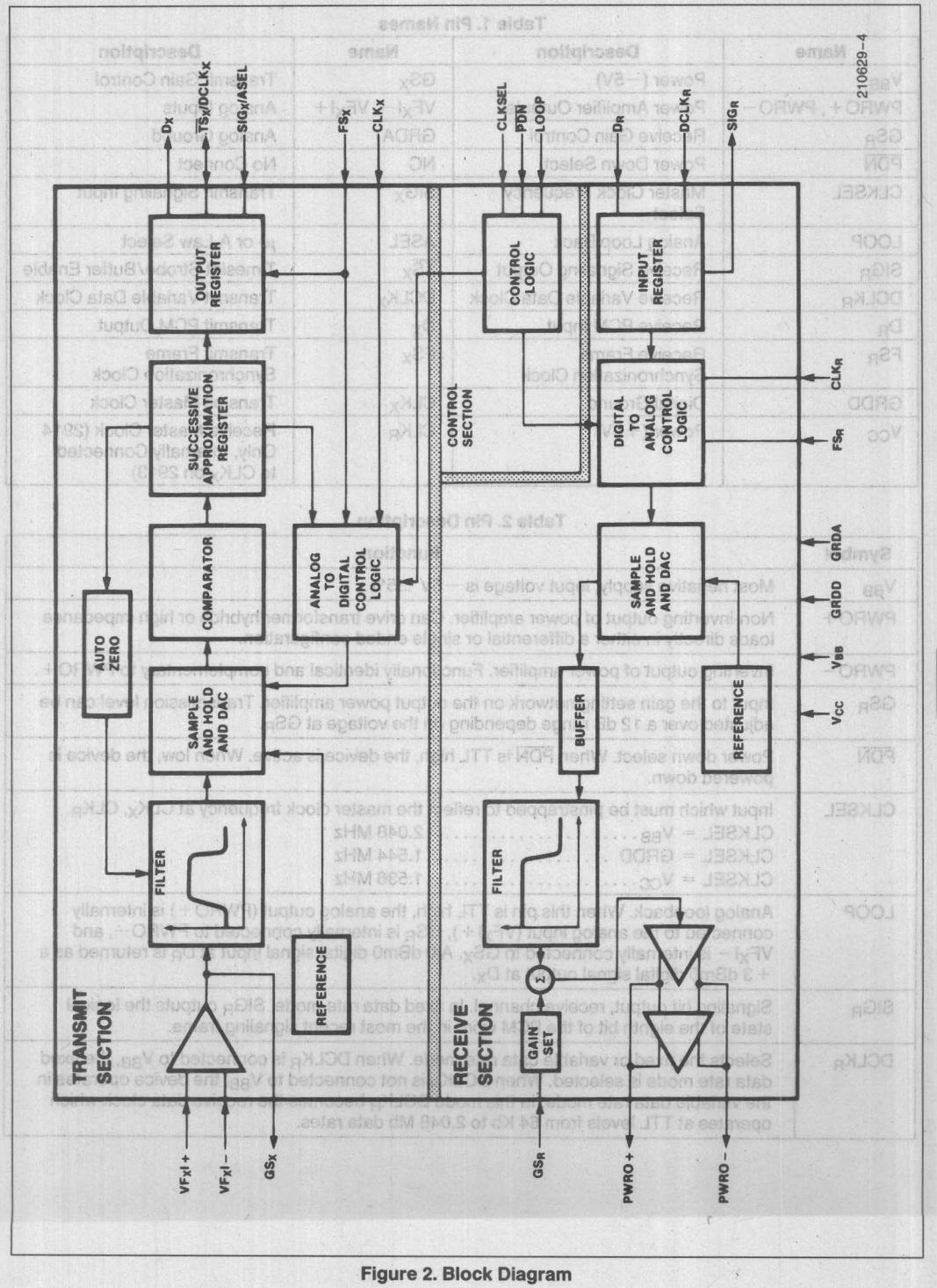


Figure 2. Block Diagram

Table 1. Pin Names

Name	Description	Name	Description
V_{BB}	Power (−5V)	GS_X	Transmit Gain Control
PWRO+, PWRO−	Power Amplifier Outputs	VF_XI-, VF_XI+	Analog Inputs
GS_R	Receive Gain Control	GRDA	Analog Ground
$\overline{PDN}$	Power Down Select	NC	No Connect
CLKSEL	Master Clock Frequency Select	SIG_X	Transmit Signaling Input
LOOP	Analog Loop Back	ASEL	μ- or A-Law Select
SIG_R	Receive Signaling Output	$\overline{TS_X}$	Timeslot Strobe/Buffer Enable
$DCLK_R$	Receive Variable Data Clock	$DCLK_X$	Transmit Variable Data Clock
D_R	Receive PCM Input	D_X	Transmit PCM Output
FS_R	Receive Frame Synchronization Clock	FS_X	Transmit Frame Synchronization Clock
GRDD	Digital Ground	CLK_X	Transmit Master Clock
V_{CC}	Power (+5V)	CLK_R	Receive Master Clock (2914 Only, Internally Connected to CLK_X on 2913)

Table 2. Pin Description

Symbol	Function
V_{BB}	Most negative supply; input voltage is −5V ±5%.
PWRO+	Non-inverting output of power amplifier. Can drive transformer hybrids or high impedance loads directly in either a differential or single ended configuration.
PWRO−	Inverting output of power amplifier. Functionally identical and complementary to PWRO+.
GS_R	Input to the gain setting network on the output power amplifier. Transmission level can be adjusted over a 12 dB range depending on the voltage at GS_R.
$\overline{PDN}$	Power down select. When $\overline{PDN}$ is TTL high, the device is active. When low, the device is powered down.
CLKSEL	Input which must be pinstrapped to reflect the master clock frequency at CLK_X, CLK_R. CLKSEL = V_{BB} ..... 2.048 MHz CLKSEL = GRDD ..... 1.544 MHz CLKSEL = V_{CC} ..... 1.536 MHz
LOOP	Analog loopback. When this pin is TTL high, the analog output (PWRO+) is internally connected to the analog input (VF_XI+), GS_R is internally connected to PWRO−, and VF_XI- is internally connected to GS_X. A 0 dBm0 digital signal input at D_R is returned as a +3 dBm0 digital signal output at D_X.
SIG_R	Signaling bit output, receive channel. In fixed data rate mode, SIG_R outputs the logical state of the eighth bit of the PCM word in the most recent signaling frame.
$DCLK_R$	Selects the fixed or variable data rate mode. When $DCLK_R$ is connected to V_{BB}, the fixed data rate mode is selected. When $DCLK_R$ is not connected to V_{BB}, the device operates in the variable data rate mode. In this mode $DCLK_R$ becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.

Table 2. Pin Description (Continued)

Symbol	Function
D_R	Receive PCM input. PCM data is clocked in on this lead on eight consecutive negative transitions of the receive data clock; CLK_R in the fixed data rate mode and $DCLK_R$ in variable data rate mode.
FS_R	8 KHz frame synchronization clock input/timeslot enable, receive channel. A multi-function input which in fixed data rate mode distinguishes between signaling and non-signaling frames by means of a double or single wide pulse respectively. In variable data rate mode this signal must remain high for the entire length of the timeslot. The receive channel enters the standby state whenever FS_R is TTL low for 300 milliseconds.
GRDD	Digital ground for all internal logic circuits. Not internally tied to GRDA.
CLK_R	Receive master and data clock for the fixed data rate mode; receive master clock only in variable data rate mode.
CLK_X	Transmit master and data clock for the fixed data rate mode; transmit master clock only in variable data rate mode.
FS_X	8 KHz frame synchronization clock input/timeslot enable, transmit channel. Operates independently but in an analogous manner to FS_R. The transmit channel enters the standby state whenever FS_X is TTL low for 300 milliseconds.
D_X	Transmit PCM output. PCM data is clocked out on this lead on eight consecutive positive transitions of the transmit data clock: CLK_X in fixed data rate mode and $DCLK_X$ in variable data rate mode.
$\overline{TS}_X/DCLK_X$	Transmit channel timeslot strobe (output) or data clock (input) for the transmit channel. In fixed data rate mode, this pin is an open drain output designed to be used as an enable signal for a three-state buffer. In variable data rate mode, this pin becomes the transmit data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.
SIG_X/ASEL	A dual purpose pin. When connected to V_{BB}, A-law operation is selected. When it is not connected to V_{BB} this pin is a TTL level input for signaling operation. This input is transmitted as the eighth bit of the PCM word during signaling frames on the D_X lead. If not used as an input pin, ASEL should be strapped to either V_{CC} or GRDD.
NC	No connect.
GRDA	Analog ground return for all internal voice circuits. Not internally connected to GRDD.
VF_XI+	Non-inverting analog input to uncommitted transmit operational amplifier.
VF_XI-	Inverting analog input to uncommitted transmit operational amplifier.
GS_X	Output terminal of transmit channel input op amp. Internally, this is the voice signal input to the transmit filter.
V_{CC}	Most positive supply; input voltage is $+5V \pm 5\%$.

5

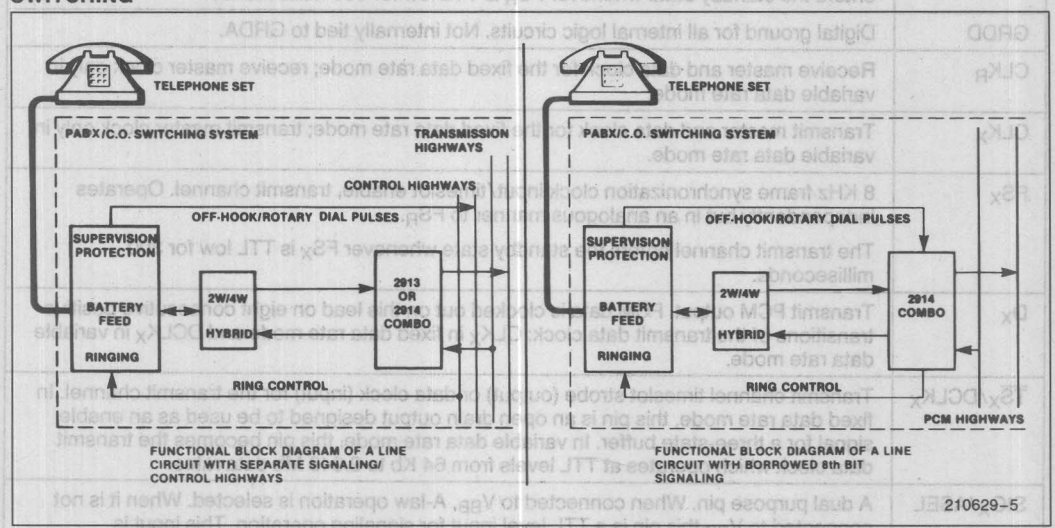
FUNCTIONAL DESCRIPTION

The 2913 and 2914 provide the analog-to-digital and the digital-to-analog conversions and the transmit and receive filtering necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. They are intended to be used at the analog termination of a PCM line or trunk.

The following major functions are provided:

- Bandpass filtering of the analog signals prior to encoding and after decoding
- Encoding and decoding of voice and call progress information
- Encoding and decoding of the signaling and supervision information

SWITCHING



CHANNEL BANKS

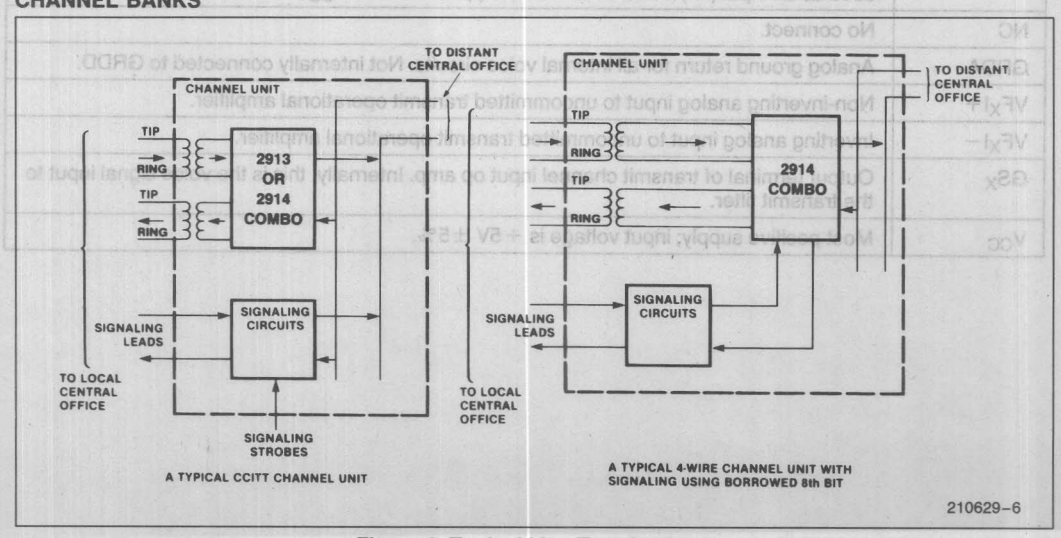


Figure 3. Typical Line Terminations

GENERAL OPERATION

System Reliability Features

The combochip can be powered up by pulsing FS_X and/or FS_R while a TTL high voltage is applied to $\overline{PDN}$, provided that all clocks and supplies are connected. The 2913 and 2914 have internal resets on power up (or when V_{BB} or V_{CC} are re-applied) in order to ensure validity of the digital outputs and thereby maintain integrity of the PCM highway.

On the transmit channel, digital outputs D_X and $\overline{TS}_X$ are held in a high impedance state for approximately four frames (500 μs) after power up or application of V_{BB} or V_{CC}. After this delay, D_X, $\overline{TS}_X$, and signaling will be functional and will occur in the proper timeslot. The analog circuits on the transmit side require approximately 60 milliseconds to reach their equilibrium value due to the autozero circuit settling time. Thus, valid digital information, such as for on/off hook detection, is available almost immediately, while analog information is available after some delay.

On the receive channel, the digital output SIG_R is also held low for a maximum of four frames after power up or application of V_{BB} or V_{CC}. SIG_R will remain low thereafter until it is updated by a signaling frame.

To further enhance system reliability, $\overline{TS}_X$ and D_X will be placed in a high impedance state approximately 30 μs after an interruption of CLK_X. Similarly, SIG_R will be held low approximately 30 μs after an interruption of CLK_R. These interruptions could possibly occur with some kind of fault condition.

Power Down and Standby Modes

To minimize power consumption, two power down modes are provided in which most 2913/2914 functions are disabled. Only the power down, clock, and frame sync buffers, which are required to power up the device, are enabled in these modes. As shown in Table 3, the digital outputs on the appropriate channels are placed in a high impedance state until the device returns to the active mode.

The Power Down mode utilizes an external control signal to the $\overline{PDN}$ pin. In this mode, power consumption is reduced to the value shown in Table 3. The device is active when the signal is high and inactive when it is low. In the absence of any signal, the $\overline{PDN}$ pin floats to TTL high allowing the device to remain active continuously.

The Standby mode leaves the user an option of powering either channel down separately or powering the entire device down by selectively removing FS_X and/or FS_R. With both channels in the standby state, power consumption is reduced to the value shown in Table 3. If transmit only operation is desired, FS_X should be applied to the device while FS_R is held low. Similarly, if receive only operation is desired, FS_R should be applied while FS_X is held low.

Fixed Data Rate Mode

Fixed data rate timing, which is 2910A and 2911A compatible, is selected by connecting $DCLK_R$ to V_{BB}. It employs master clocks CLK_X and CLK_R, frame synchronization clocks FS_X and FS_R, and output $\overline{TS}_X$.

5

Table 3. Power-Down Methods

Device Status	Power-Down Method	Typical Power Consumption	Digital Output Status
Power Down Mode	$\overline{PDN}$ = TTL Low	5 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state and SIG_R is placed in a TTL low state within 10 μs.
Standby Mode	FS_X and FS_R are TTL Low	12 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state and SIG_R is placed in a TTL low state 300 milliseconds after FS_X and FS_R are removed.
Only Transmit Is on Standby	FS_X is TTL Low	70 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state within 300 milliseconds.
Only Receive Is on Standby	FS_R is TTL Low	110 mW	SIG_R is placed in a TTL low state within 300 milliseconds.

CLK_X and CLK_R serve both as master clocks to operate the codec and filter sections and bit clocks to clock the data in and out from the PCM highway. FS_X and FS_R are 8 KHz inputs which set the sampling frequency and distinguish between signaling and non-signaling frames by their pulse width. A frame synchronization pulse which is one master clock wide designates a non-signaling frame, while a double wide sync pulse enables the signaling function. $\overline{TS}_X$ is a timeslot strobe/buffer enable output which gates the PCM word onto the PCM highway when an external buffer is used to drive the line.

Data is transmitted on the highway at D_X on the first eight positive transitions of CLK_X following the rising edge of FS_X. Similarly, on the receive side, data is received on the first eight falling edges of CLK_R. The frequency of CLK_X and CLK_R is selected by the CLKSEL pin to be either 1.536, 1.544, or 2.048 MHz. No other frequency of operation is allowed in the fixed data rate mode.

Variable Data Rate Mode

Variable data rate timing is selected by connecting $DCLK_R$ to the bit clock for the receive PCM highway rather than to V_{BB}. It employs master clocks CLK_X and CLK_R, bit clocks $DCLK_R$ and $DCLK_X$, and frame synchronization clocks FS_R and FS_X.

Variable data rate timing allows for a flexible data frequency. It provides the ability to vary the frequency of the bit clocks, which can be asynchronous in the case of the 2914 or synchronous in the case of the 2913, from 64 KHz to 2.048 MHz. Master clock's inputs are still restricted to 1.536, 1.544, or 2.048 MHz.

In this mode, $DCLK_R$ and $DCLK_X$ become the data clocks for the receive and transmit PCM highways. While FS_X is high, PCM data from D_X is transmitted onto the highway on the next eight consecutive positive transitions of $DCLK_X$. Similarly, while FS_R is high, each PCM bit from the highway is received by D_R on the next eight consecutive negative transitions of $DCLK_R$.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 μs frame as long as $DCLK_X$ is pulsed and FS_X is held high. This feature allows the PCM word to be transmitted to the PCM highway more than once per frame, if desired, and is only available in the variable data rate mode. Conversely, signaling is only allowed in the fixed data rate mode since the variable mode provides no means with which to specify a signaling frame.

Signaling

Signaling can only be performed with the 24-pin device in the fixed data rate timing mode ($DCLK_R = V_{BB}$). Signaling frames on the transmit and receive sides are independent of one another and are selected by a double-width frame sync pulse on the appropriate channel. During a transmit signaling frame, the codec will encode the incoming analog signal and substitute the signal present on SIG_X for the least significant bit of the encoded PCM word. Similarly, in a receive signaling frame, the codec will decode the seven most significant bits according to CCITT recommendation G.733 and output the logical state of the LSB on the SIG_R lead until it is updated in the next signaling frame. Timing relationships for signaling operation are shown in Figure 4.

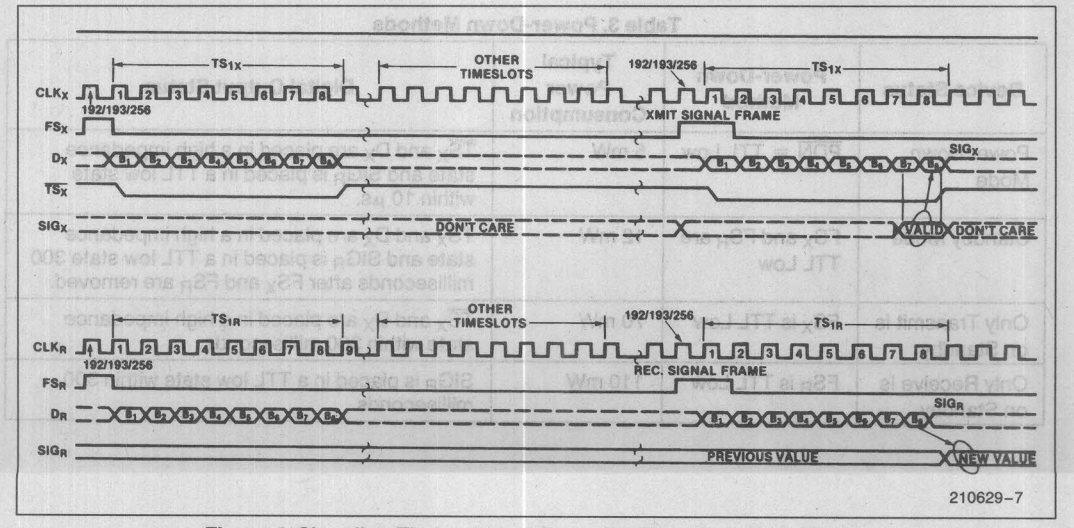


Figure 4. Signaling Timing (Used Only with Fixed Data Rate Mode)

Asynchronous Operation

The 2914 can be operated with asynchronous clocks in either the fixed or variable data rate modes. In order to avoid crosstalk problems associated with special interrupt circuitry, the design of the Intel 2913/2914 combochip includes separate digital-to-analog converters and voltage references on the transmit and receive sides to allow independent operation of the two channels.

In either timing mode, the master clock, data clock, and timeslot strobe must be synchronized at the beginning of each frame. CLK_X and $DCLK_X$ are synchronized once per frame but may be of different frequencies. The receive channel operates in a similar manner and is completely independent of the transmit timing (refer to Variable Data Rate Timing Diagrams). This approach requires the provision of two separate master clocks, even in variable data rate mode, but avoids the use of a synchronizer which can cause intermittent data conversion errors.

Analog Loopback

A distinctive feature of the 2914 is its analog loopback capability. This feature allows the user to send a control signal which internally connects the analog input and output ports. As shown in Figure 5, when LOOP is TTL high the analog output (PWRO+) is internally connected to the analog input (VF_XI+), GS_R is internally connected to PWRO−, and VF_XI- is internally connected to GS_X.

With this feature, the user can test the line circuit remotely by comparing the digital codes sent into the receive channel (D_R) with those generated on the transmit channel (D_X). Due to the difference in transmission levels between the transmit and receive sides, a 0 dBm0 code sent into D_R will emerge from D_X as a +3 dBm0 code, an implicit gain of 3 dB. Thus, the maximum signal input level which can be tested using analog loopback is 0 dBm0.

Precision Voltage References

No external components are required with the combochip to provide the voltage reference function. Voltage references are generated on-chip and are calibrated during the manufacturing process. The technique uses a difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature and bias stable reference voltage. These references determine the gain and dynamic range characteristics of the device.

Separate references are supplied to the transmit and receive sections and each is trimmed independently during the manufacturing process. The reference value is then further trimmed in the gain setting op-amps to a final precision value. With this method the combochip can achieve the extremely accurate Digital Milliwatt Responses specified in the transmission parameters, providing the user a significant margin for error in other board components.

Conversion Laws

The 2913 and 2914 are designed to operate in both μ-law and A-law systems. The user can select either conversion law according to the voltage present on the SIG_X/ASEL pin. In each case the coder and decoder process a companded 8-bit PCM word following CCITT recommendation G.711 for μ-law and A-law conversion. If A-law operation is desired, SIG_X should be tied to V_{BB}. Thus, signaling is not allowed during A-law operation. If μ = 255-law operation is selected, then SIG_X is a TTL level input which modifies the LSB of the PCM output in signaling frames.

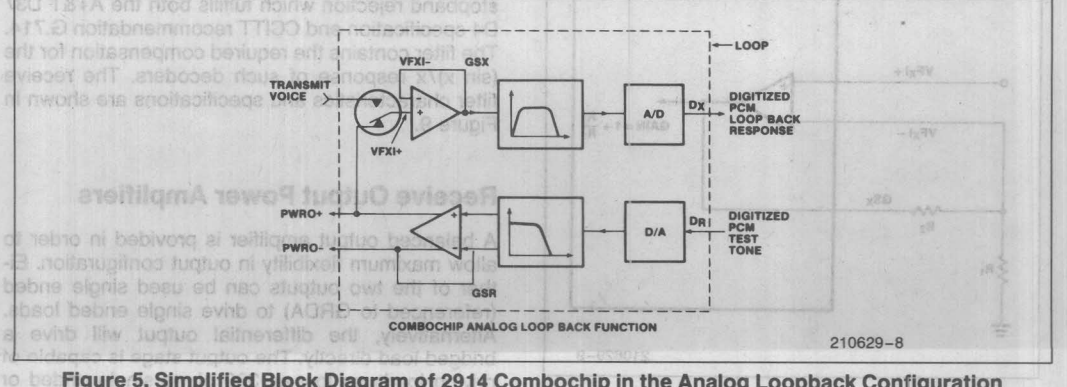


Figure 5. Simplified Block Diagram of 2914 Combochip in the Analog Loopback Configuration

VF_XI+ can be either AC or DC coupled. The input op amp can also be used in the inverting mode or differential amplifier mode (see Figure 6).

A low pass anti-aliasing section is included on-chip. This section typically provides 35 dB attenuation at the sampling frequency. No external components are required to provide the necessary anti-aliasing function for the switched capacitor section of the transmit filter.

The passband section provides flatness and stopband attenuation which fulfills the AT&T D3/D4 channel bank transmission specification and CCITT recommendation G.714. The 2913 and 2914 specifications meet or exceed digital class 5 central office switching systems requirements. The transmit filter transfer characteristics and specifications will be within the limits shown in Figure 8.

A high pass section configuration was chosen to reject low frequency noise from 50 Hz and 60 Hz power lines, 17 Hz European electric railroads, ringing

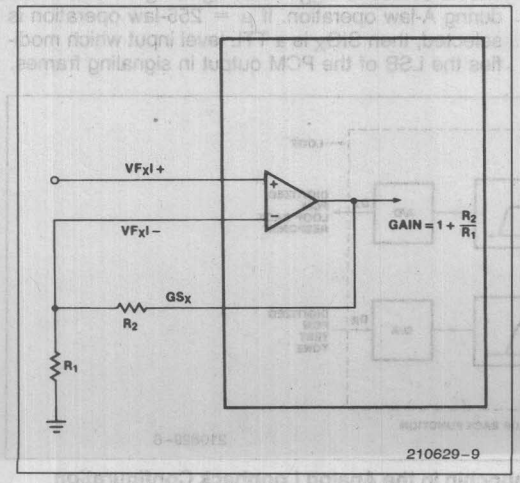


Figure 6. Transmit Filter Gain Adjustment

is transmitted on the first eight data clock bits of the next frame.

An on-chip autozero circuit corrects for DC-offset on the input signal to the encoder. This autozero circuit uses the sign bit averaging technique; the sign bit from the encoder output is long term averaged and subtracted from the input to the encoder. In this way, all DC offset is removed from the encoder input waveform.

RECEIVE OPERATION

Decoding

The PCM word at the D_R lead is serially fetched on the first eight data clock bits of the frame. A D/A conversion is performed on the digital word and the corresponding analog sample is held on an internal sample and hold capacitor. This sample is then transferred to the receive filter.

Receive Filter

The receive filter provides passband flatness and stopband rejection which fulfills both the AT&T D3/D4 specification and CCITT recommendation G.714. The filter contains the required compensation for the (sin x)/x response of such decoders. The receive filter characteristics and specifications are shown in Figure 9.

Receive Output Power Amplifiers

A balanced output amplifier is provided in order to allow maximum flexibility in output configuration. Either of the two outputs can be used single ended (referenced to GRDA) to drive single ended loads. Alternatively, the differential output will drive a bridged load directly. The output stage is capable of driving loads as low as 300 ohms single ended or 600 ohms differentially.

Table 4. Zero Transmission Level Points

Symbol	Parameter	Value	Units	Test Conditions
$0TLP1_X$	Zero Transmission Level Point Transmit Channel (0 dBm0) μ-Law	+2.76 +1.00	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP2_X$	Zero Transmission Level Point Transmit Channel (0 dBm0) A-Law	+2.79 +1.03	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP1_R$	Zero Transmission Level Point Receive Channel (0 dBm0) μ-Law	+5.76 +4.00	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP2_R$	Zero Transmission Level Point Receive Channel (0 dBm0) A-Law	+5.79 +4.03	dBm dBm	Referenced to 600Ω Referenced to 900Ω

The receive channel transmission level may be adjusted between specified limits by manipulation of the GS_R input. GS_R is internally connected to an analog gain setting network. When GS_R is strapped to PWRO−, the receive level is maximized; when it is tied to PWRO+, the level is minimized. The output transmission level interpolates between 0 dB and −12 dB as GS_R is interpolated (with a potentiometer) between PWRO+ and PWRO−. The use of the output gain set is illustrated in Figure 7.

Transmission levels are specified relative to the receive channel output under digital milliwatt conditions, that is, when the digital input at D_R is the eight-code sequence specified in CCITT recommendation G.711.

OUTPUT GAIN SET: DESIGN CONSIDERATIONS

(Refer to Figure 7)

PWRO+ and PWRO− are low impedance complementary outputs. The voltages at the nodes are:

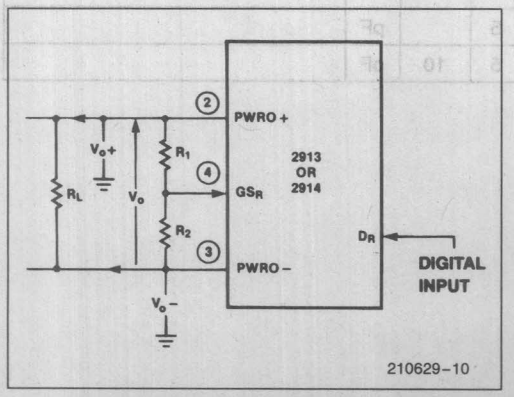


Figure 7. Gain Setting Configuration

V_O+ at PWRO+

V_O− at PWRO−

$V_O = (V_O+) - (V_O-)$(total differential response)

R_1 and R_2 are a gain setting resistor network with the center tap connected to the GS_R input.

A value greater than 10K ohms for $R_1 + R_2$ and less than 100K ohms for R_1 in parallel with R_2 is recommended because:

(a) The parallel combination of $R_1 + R_2$ and R_L sets the total loading.

(b) The total capacitance at the GS_R input and the parallel combination of R_1 and R_2 define a time constant which has to be minimized to avoid inaccuracies.

A is the gain of the power amplifiers, where

$$A = \frac{1 + (R_1/R_2)}{4 + (R_1/R_2)}$$

For design purposes, a useful form is R_1/R_2 as a function of A.

$$R_1/R_2 = \frac{4A - 1}{1 - A}$$

(Allowable values for A are those which make R_1/R_2 positive.)

Examples are:

If A = 1 (maximum output), then

$$R_1/R_2 = \infty \text{ or } V(GS_R) = V_O-;$$

i.e., GS_R is tied to PWRO−

If A = ½, then

$$R_1/R_2 = 2$$

If A = ¼, (minimum output) then

$$R_1/R_2 = 0 \text{ or } V(GS_R) = V_O+;$$

i.e., GS_R is tied to PWRO+.

5

ABSOLUTE MAXIMUM RATINGS

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

Temperature under Bias ..... −10°C to +80°C
Storage Temperature ..... −65°C to +150°C
V_{CC} and GRDD with Respect to V_{BB} ..... −0.3V to +15V
All Input and Output Voltages with Respect to V_{BB} ..... −0.3V to +15V
Power Dissipation 1.35W

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

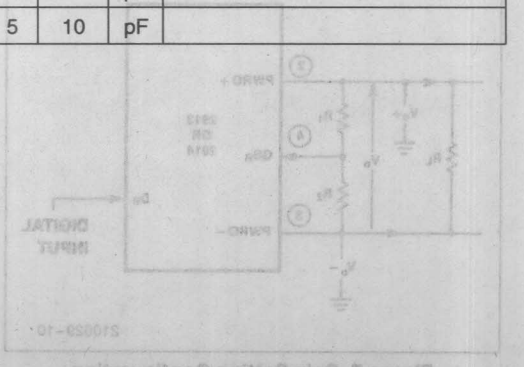
D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified

Typical values are for T_A = 25°C and nominal power supply values

DIGITAL INTERFACE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{IL}	Low Level Input Current			10	μA	GRDD ≤ V_{IN} ≤ V_{IL}(1)
I_{IH}	High Level Input Current			10	μA	V_{IH} ≤ V_{IN} ≤ V_{CC}
V_{IL}	Input Low Voltage, except CLKSEL			0.8	V	
V_{IH}	Input High Voltage, except CLKSEL	2.0			V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 3.2 mA at D_X, $\overline{TS}_X$ and SIG_R
V_{OH}	Output High Voltage	2.4			V	I_{OH} = 9.6 mA at D_X I_{OH} = 1.2 mA at SIG_R
V_{ILO}	Input Low Voltage, CLKSEL(2)	V_{BB}		V_{BB} + 0.5	V	
V_{IIO}	Input Intermediate Voltage, CLKSEL	GRDD − 0.5		0.5	V	
V_{IHO}	Input High Voltage, CLKSEL	V_{CC} − 0.5		V_{CC}	V	
C_{OX}	Digital Output Capacitance(3)		5		pF	
C_{IN}	Digital Input Capacitance		5	10	pF	



D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified

Typical values are for T_A = 25°C and nominal power supply values (Continued)

POWER DISSIPATION All measurements made at f_{DCLK} = 2.048 MHz, outputs unloaded

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{CC1}	V_{CC} Operating Current(5)		14	19	mA	
I_{BB1}	V_{BB} Operating Current		−18	−24	mA	
I_{CC0}	V_{CC} Power Down Current		0.5	1.0	mA	$\overline{PDN} \leq V_{IL}$; after 10 μs
I_{BB0}	V_{BB} Power Down Current		−0.5	−1.0	mA	$\overline{PDN} \leq V_{IL}$; after 10 μs
I_{CCS}	V_{CC} Standby Current		1.2	2.4	mA	FS_X, $FS_R \leq V_{IL}$; after 300 ms
I_{BBS}	V_{BB} Standby Current		−1.2	−2.4	mA	FS_X, $FS_R \leq V_{IL}$; after 300 ms
P_{D1}	Operating Power Dissipation(4)		140	200	mW	
P_{D0}	Power Down Dissipation(4)		5	10	mW	$\overline{PDN} \leq V_{IL}$; after 10 μs
P_{ST}	Standby Power Dissipation(4)		12	25	mW	FS_X, $FS_R \leq V_{IL}$

NOTES:
1. V_{IN} is the voltage on any digital pin.
2. SIG_X and $DCLK_R$ are TTL level inputs between GRDD and V_{CC}; they are also pin straps for mode selection when tied to V_{BB}. Under these conditions V_{ILO} is the input low voltage requirement.
3. Timing parameters are guaranteed based on a 100 pF load capacitance. Up to eight digital outputs may be connected to a common PCM highway without buffering, assuming a board capacitance of 60 pF.
4. With nominal power supply values.
5. V_{CC} applied last or simultaneously with V_{BB}.

5

ANALOG INTERFACE, TRANSMIT CHANNEL INPUT STAGE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{BXI}	Input Leakage Current, VF_XI+, VF_XI-			100	nA	$-2.17V \leq V_{IN} \leq 2.17V$
R_{IXI}	Input Resistance, VF_XI+, VF_XI-	10			MΩ	
V_{OSXI}	Input Offset Voltage, VF_XI+, VF_XI-			25	mV	
CMRR	Common Mode Rejection, VF_XI+, VF_XI-	55			dB	$-2.17V \leq V_{IN} \leq 2.17V$
A_{VOL}	DC Open Loop Voltage Gain, GS_X	5000				
f_C	Open Loop Unity Gain Bandwidth, GS_X		1		MHz	
C_{LXI}	Load Capacitance, GS_X			50	pF	
R_{LXI}	Minimum Load Resistance, GS_X	10			KΩ	

ANALOG INTERFACE, RECEIVE CHANNEL DRIVER AMPLIFIER STAGE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
R_{ORA}	Output Resistance, PWRO+, PWRO−		1		Ω	
V_{OSRA}	Single-Ended Output DC Offset, PWRO+, PWRO−		75	±150	mV	Relative to GRDA
C_{LRA}	Load Capacitance, PWRO+, PWRO−			100	pF	

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS

Unless otherwise noted, the analog input is a 0 dBm0, 1020 Hz sine wave.(1) Input amplifier is set for unity gain, noninverting. The digital input is a PCM bit stream generated by passing a 0 dBm0, 1020 Hz sine wave through an ideal encoder. Receive output is measured single ended, maximum gain configuration.(2) All output levels are (sin x)/x corrected. Specifications are for synchronous operation. Typical values are for $T_A = 25°C$ and nominal power supply values. ($T_A = 0°C$ to $+70°C$; $V_{CC} = +5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified).

GAIN AND DYNAMIC RANGE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
EmW	Encoder Milliwatt Response Tolerance	−0.18	±0.04	+0.18	dBm0	Signal input of 1.064 Vrms μ-law Signal input of 1.068 Vrms A-law $T_A = 25°C$, $V_{BB} = -5V$, $V_{CC} = +5V$
EmW_{TS}	EmW Variation with Temperature and Supplies	−0.07	±0.02	+0.07	dB	±5% supplies, 0 to 70°C Relative to nominal conditions
DmW	Digital Milliwatt Response Tolerance	−0.18	±0.04	+0.18	dBm0	Measure relative to $0TLP_R$. Signal input per CCITT Recommendation G.711. Output signal of 1000 Hz, $R_L = \infty$; $T_A = 25°C$; $V_{BB} = -5V$, $V_{CC} = +5V$.
DmW_{TS}	DmW Variation with Temperature and Supplies	−0.07	+0.02	+0.07	dB	±5% supplies, 0 to 70°C

NOTES:
1. 0 dBm0 is defined as the zero reference point of the channel under test (0TLP). This corresponds to an analog signal input of 1.064 Vrms or an output of 1.503 Vrms for μ-law. See Table 4.
2. Unity gain input amplifier: GS_X is connected to VF_XI-, Signal input VF_XI+; Maximum gain output amplifier; GS_R is connected to PWRO−, output to PWRO+.

GAIN TRACKING Reference Level = −10 dBm0

Symbol	Parameter	2913-1, 2914-1		2913, 2914		Unit	Test Conditions
		Min	Max	Min	Max		
$GT1_X$	Transmit Gain Tracking Error Sinusoidal Input; μ-Law		±0.2		±0.25	dB	+3 to −40 dBm0
			±0.3		±0.5	dB	−40 to −50 dBm0
			±0.65		±1.2	dB	−50 to −55 dBm0
$GT2_X$	Transmit Gain Tracking Error Sinusoidal Input; A-Law		±0.2		±0.25	dB	+3 to −40 dBm0
			±0.3		±0.5	dB	−40 to −50 dBm0
			±0.65		±1.2	dB	−50 to −55 dBm0
$GT1_R$	Receive Gain Tracking Error Sinusoidal Input; μ-Law		±0.2		±0.25	dB	+3 to −40 dBm0
			±0.3		±0.5	dB	−40 to −50 dBm0
			±0.65		±1.2	dB	−50 to −55 dBm0
							Measured at PWRO+, $R_L = 300\Omega$
$GT2_R$	Receive Gain Tracking Error Sinusoidal Input; A-Law		±0.2		±0.25	dB	+3 to −40 dBm0
			±0.3		±0.5	dB	−40 to −50 dBm0
			±0.65		±1.2	dB	−50 to −55 dBm0
							Measured at PWRO+, $R_L = 300\Omega$

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

NOISE All receive channel measurements are single ended

Symbol	Parameter	2913-1, 2914-1			2913, 2914			Unit	Test Conditions
		Min	Typ	Max	Min	Typ	Max		
N_{XC1}	Transmit Noise, C-Message Weighted			13			15	dBrnc0	VF_XI+ = GRDA, VF_XI- = GS_X
N_{XC2}	Transmit Noise, C-Message Weighted with Eighth Bit Signaling			16			18	dBrnc0	VF_XI+ = GRDA, VF_XI- = GS_X; 6th Frame Signaling
N_{XP}	Transmit Noise, Psophometrically Weighted			−77			−75	dBm0p	VF_XI+ = GRDA, VF_XI- = GS_X
N_{RC1}	Receive Noise, C-Message Weighted: Quiet Code			8			11	dBrnc0	D_R = 11111111
N_{RC2}	Receive Noise, C-Message Weighted: Sign Bit Toggle			9			12	dBrnc0	Input to D_R is zero code with sign bit toggle at 1 KHz rate
N_{RP}	Receive Noise, Psophometrically Weighted			−82			−79	dBm0p	D_R = lowest positive decode level
N_{SF}	Single Frequency Noise End to End Measurement			−50			−50	dBm0	CCITT G.712.4.2, measure at PWRO+
$PSRR_1$	V_{CC} Power Supply Rejection, Transmit Channel		−30			−30		dB	Idle channel; 200 mV P-P signal on supply; 0 to 50 KHz, measure at D_X
$PSRR_2$	V_{BB} Power Supply Rejection, Transmit Channel		−30			−30		dB	Idle channel; 200 mV P-P signal on supply; 0 to 50 KHz, measure at D_X
$PSRR_3$	V_{CC} Power Supply Rejection, Receive Channel		−25			−25		dB	Idle channel; 200 mV P-P signal on supply; measure narrow band at PWRO+, 0 to 50 KHz
$PSRR_4$	V_{BB} Power Supply Rejection, Receive Channel		−25			−25		dB	Idle channel; 200 mV P-P signal on supply; measure narrow band at PWRO+, 0 to 50 KHz
CT_{TR}	Crosstalk, Transmit to Receive			−80			−71	dB	VF_XI+ = 0 dBm0, 1.02 KHz, D_R = lowest positive decode level, measure at PWRO+
CT_{RT}	Crosstalk, Receive to Transmit			−80			−71	dB	D_R = 0 dBm0, 1.02 KHz VF_XI+ = GRDA, measure at D_X

5

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

DISTORTION

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$SD1_X$	Transmit Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2	36 30 25			dB dB dB	0 to −30 dBm0 −30 to −40 dBm0 −40 to −45 dBm0
$SD2_X$	Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2	36 30 25			dB dB dB	0 to −30 dBm0 −30 to −40 dBm0 −40 to −45 dBm0
$SD1_R$	Receive Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2	36 30 25			dB dB dB	0 to −30 dBm0 −30 to −40 dBm0 −40 to −45 dBm0
$SD2_R$	Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2	36 30 25			dB dB dB	0 to −30 dBm0 −30 to −40 dBm0 −40 to −45 dBm0
DP_X	Transmit Single Frequency Distortion Products			−46	dBm0	AT&T Advisory #64 (3.8) 0 dBm0 Input Signal
DP_R	Receive Single Frequency Distortion Products			−46	dBm0	AT&T Advisory #64 (3.8) 0 dBm0 Input Signal
IMD_1	Intermodulation Distortion, End to End Measurement			−35	dB	CCITT G.712 (7.1)
IMD_2	Intermodulation Distortion, End to End Measurement			−49	dBm0	CCITT G.712 (7.2)
SOS	Spurious Out of Band Signals, End to End Measurement			−25	dBm0	CCITT G.712 (6.1)
SIS	Spurious in Band Signals, End to End Measurement			−40	dBm0	CCITT G.712 (9)
D_{AX}	Transmit Absolute Delay		245		μs	Fixed Data Rate. CLK_X = 2.048 MHz 0 dBm0, 1.02 KHz signal at VF_XI+ Measure at D_X.
D_{DX}	Transmit Differential Envelope Delay Relative to D_{AX}		170 95 45 105		μs μs μs μs	f = 500 − 600 Hz f = 600 − 1000 Hz f = 1000 − 2600 Hz f = 2600 − 2800 Hz
D_{AR}	Receive Absolute Delay		190		μs	Fixed Data Rate, CLK_R = 2.048 MHz; Digital input is DMW codes. Measure at PWRO+.
D_{DR}	Receive Differential Envelope Delay Relative to D_{AR}		45 35 85 110		μs μs μs μs	f = 500 − 600 Hz f = 600 − 1000 Hz f = 1000 − 2600 Hz f = 2600 − 2800 Hz

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

TRANSMIT CHANNEL TRANSFER CHARACTERISTICS

Input amplifier is set for unity gain; noninverting; maximum gain output.

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
G_{RX}	Gain Relative to Gain at 1.02 KHz					0 dBm0 Signal input at VF_XI+
	16.67 Hz			−30	dB	
	50 Hz			−25	dB	
	60 Hz			−23	dB	
	200 Hz	−1.8		−0.125	dB	
	300 to 3000 Hz	−0.125		+0.125	dB	
	3300 Hz	−0.35		+0.03	dB	
	3400 Hz	−0.7		−0.10	dB	
	4000 Hz			−14	dB	
	4600 Hz and Above			−32	dB	

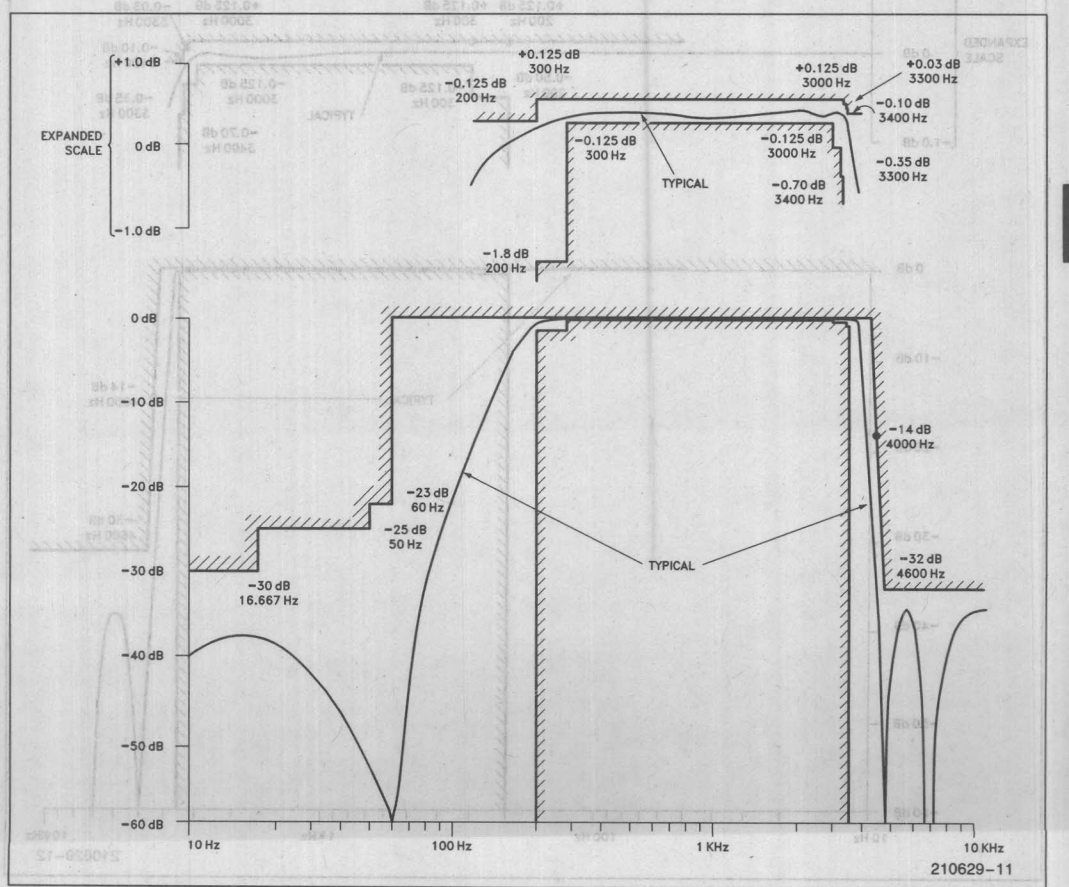


Figure 8. Transmit Channel

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS (Continued)

RECEIVE CHANNEL TRANSFER CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
G_{RR}	Gain Relative to Gain at 1.02 KHz					0 dBm0 Signal input at D_R
	Below 200 Hz			+0.125	dB	
	200 Hz	−0.5		+0.125	dB	
	300 to 3000 Hz	−0.125		+0.125	dB	
	3300 Hz	−0.35		+0.03	dB	
	3400 Hz	−0.7		−0.1	dB	
	4000 Hz			−14	dB	
	4600 Hz and Above			−30	dB	

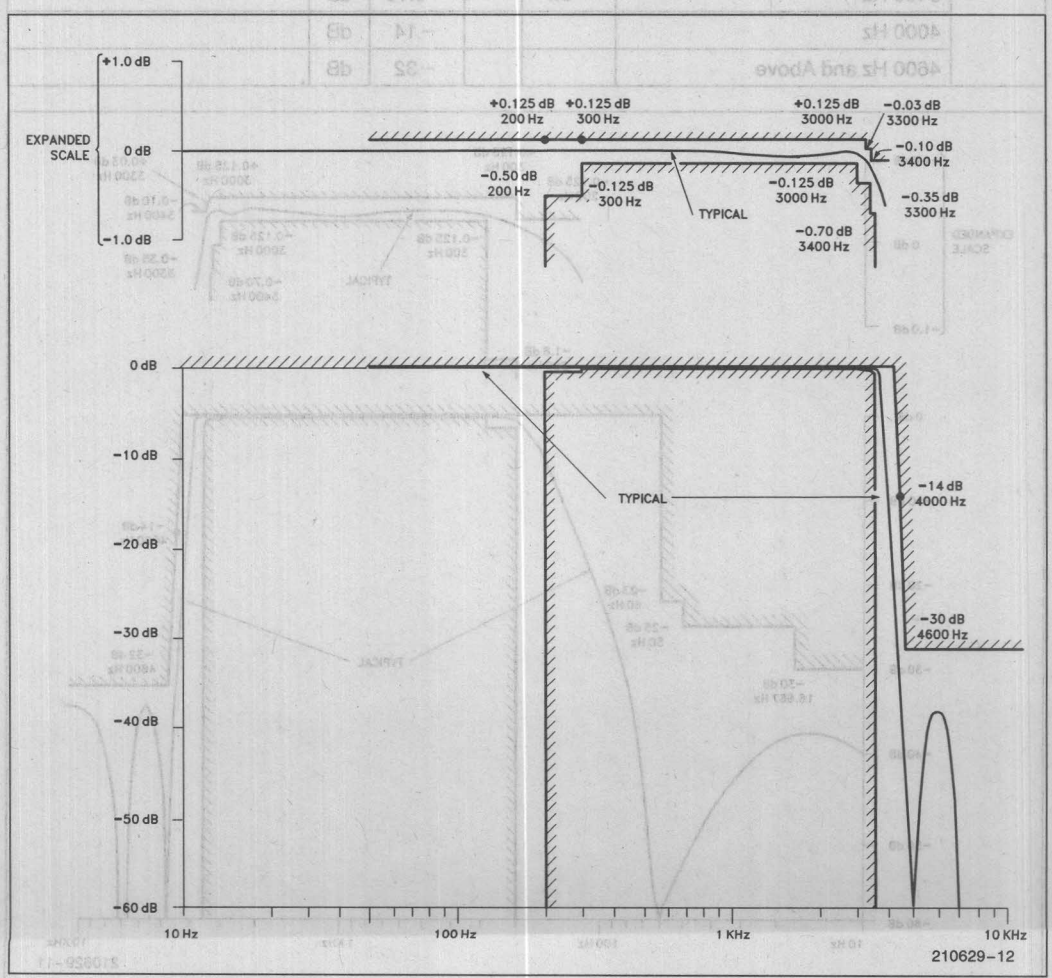


Figure 9. Receive Channel

A.C. CHARACTERISTICS—TIMING PARAMETERS

CLOCK SECTION

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{CY}	Clock Period, CLK_X, CLK_R	488			ns	$f_{CLKX} = f_{CLKR} = 2.048$ MHz
t_{CLK}	Clock Pulse Width, CLK_X, CLK_R	220			ns	
t_{DCLK}	Data Clock Pulse Width	220			ns	64 KHz $\leq f_{DCLK} \leq$ 2.048 MHz
t_{CDC}	Clock Duty Cycle, CLK_X, CLK_R	45	50	55	%	
t_r, t_f	Clock Rise and Fall Time	5		30	ns	

TRANSMIT SECTION, FIXED DATA RATE MODE(1)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{DZX}	Data Enabled on TS Entry	0		145	ns	$0 < C_{LOAD} < 100$ pF
t_{DDX}	Data Delay from CLK_X	0		145	ns	$0 < C_{LOAD} < 100$ pF
t_{HZX}	Data Float on TS Exit	60		215	ns	$C_{LOAD} = 0$
t_{SON}	Timeslot X to Enable	0		145	ns	$0 < C_{LOAD} < 100$ pF
t_{SOFF}	Timeslot X to Disable	60		215	ns	$C_{LOAD} = 0$
t_{FSD}	Frame Sync Delay	100		t_{CLK}	ns	
t_{SS}	Signal Setup Time	0			ns	
t_{SH}	Signal Hold Time	0			ns	

RECEIVE SECTION, FIXED DATA RATE MODE

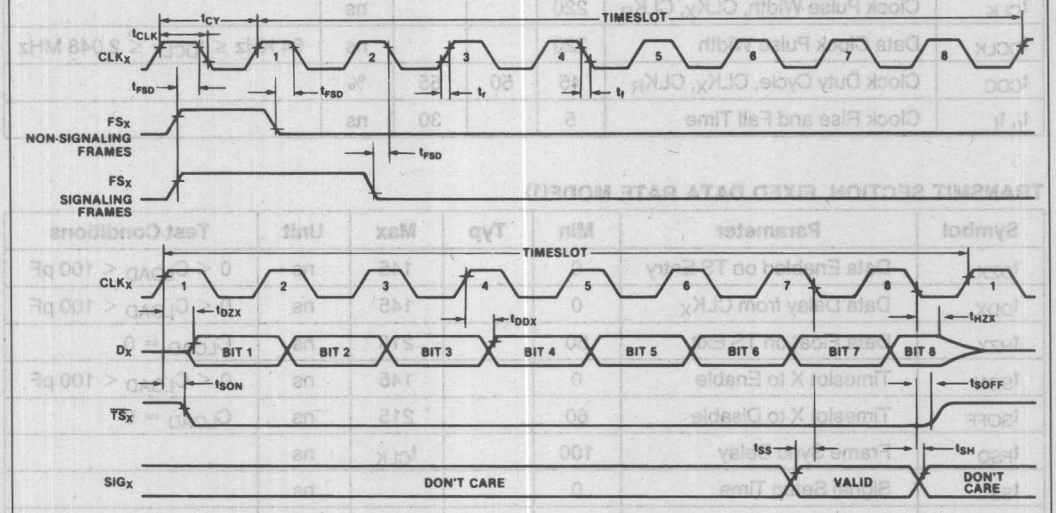
Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{DSR}	Receive Data Setup	10			ns	
t_{DHR}	Receive Data Hold	60			ns	
t_{FSD}	Frame Sync Delay	100		t_{CLK}	ns	
t_{SIGR}	SIG_R Update	0		2	μs	

NOTE:
1. Timing parameters t_{DZX}, t_{HZX}, and t_{SOFF} are referenced to a high impedance state.

WAVEFORMS

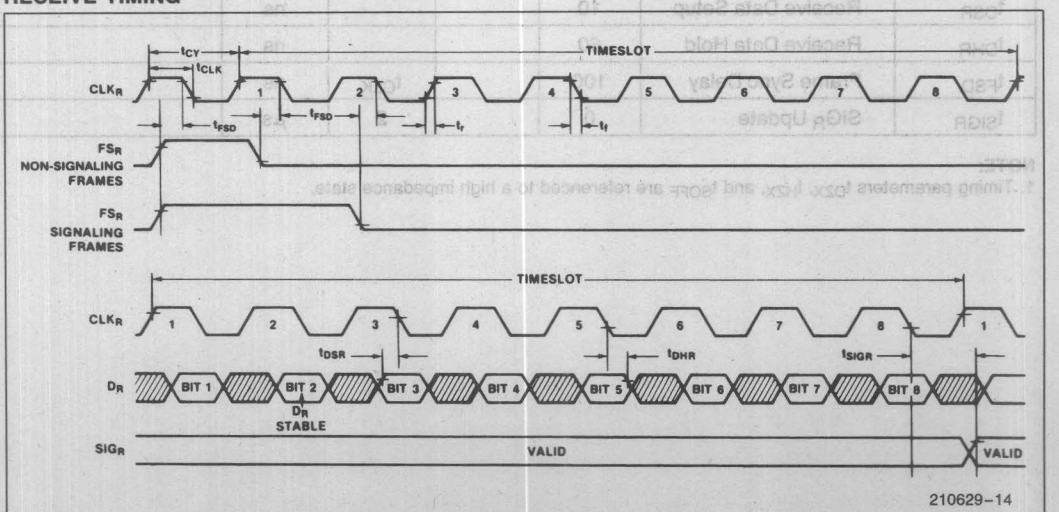
Fixed Data Rate Timing

TRANSMIT TIMING



NOTE:
All timing parameters referenced to V_{IH} and V_{IL} except t_{DZX}, t_{SOFF} and t_{HZX} which reference a high impedance state.

RECEIVE TIMING



NOTE:
All timing parameters referenced to V_{IH} and V_{IL}.

WAVEFORMS (Continued)

TRANSMIT SECTION, VARIABLE DATA RATE MODE(1)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{TSDX}	Timeslot Delay from $DCLK_X$(2)	140		$t_{DX} - 140$	ns	
t_{FSD}	Frame Sync Delay	100		$t_{CY} - 100$	ns	
t_{DDX}	Data Delay from $DCLK_X$	0		100	ns	$0 < C_{LOAD} < 100$ pF
t_{DON}	Timeslot to D_X Active	0		50	ns	$0 < C_{LOAD} < 100$ pF
t_{DOFF}	Timeslot to D_X Inactive	0		80	ns	$0 < C_{LOAD} < 100$ pF
t_{DX}	Data Clock Period	488		15620	ns	
t_{DFSX}	Data Delay from FS_X	0		140	ns	

RECEIVE SECTION, VARIABLE DATA RATE MODE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{TSDR}	Timeslot Delay from $DCLK_R$(3)	140		$t_{DR} - 140$	ns	
t_{FSD}	Frame Sync Delay	100		$t_{CY} - 100$	ns	
t_{DSR}	Data Setup Time	10			ns	
t_{DHR}	Data Hold Time	60			ns	
t_{DR}	Data Clock Period	488		15620	ns	
t_{SER}	Timeslot End Receive Time	60			ns	

64 KB OPERATION, VARIABLE DATA RATE MODE

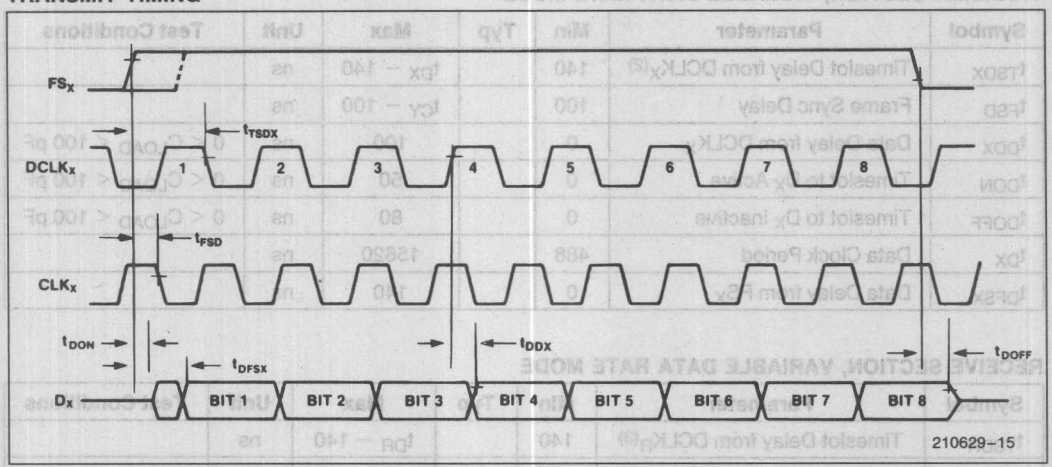
Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{FSLX}	Transmit Frame Sync Minimum Downtime	488			ns	FS_X is TTL high for remainder of frame
t_{FSLR}	Receive Frame Sync Minimum Downtime	1952			ns	FS_R is TTL high for remainder of frame
t_{DCLK}	Data Clock Pulse Width			10	μs	

NOTES:
1. Timing parameters for t_{DON} and t_{DOFF} are referenced to a high impedance state.
2. t_{FSLX} minimum requirements override t_{TSDX} maximum spec for 64 KHz operation.
3. t_{FSLR} minimum requirements override t_{TSDR} maximum spec for 64 KHz operation.

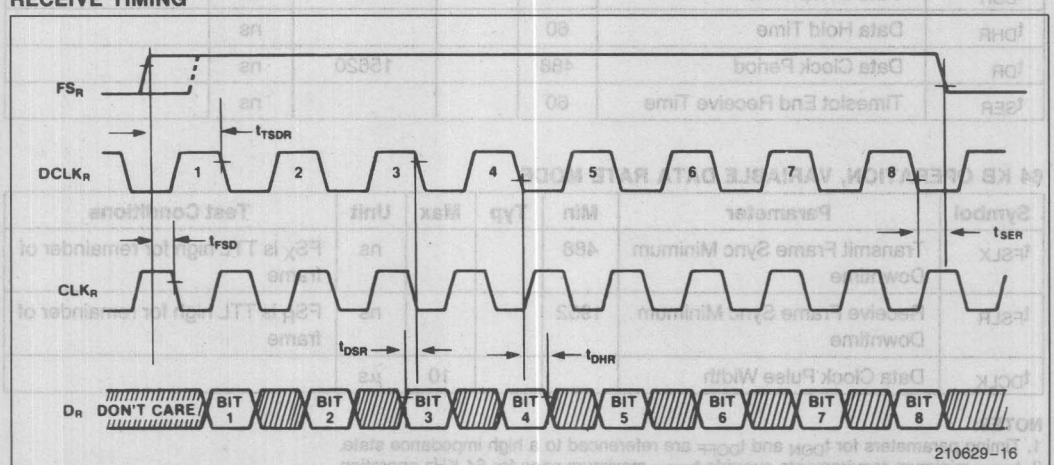
5

VARIABLE DATA RATE TIMING

TRANSMIT TIMING

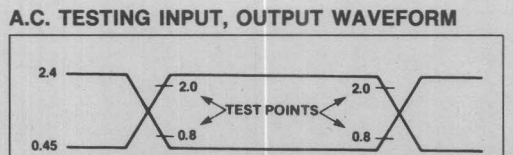


RECEIVE TIMING



NOTE:
All timing parameters referenced to V_{IH} and V_{IL} except t_{DON} and t_{DOFF} which reference a high impedance state.

A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

2916/2917
HMOS COMBINED SINGLE CHIP PCM CODEC AND FILTER

- **2916 μ-Law, 2.048 MHz Master Clock**
- **2917 A-Law, 2.048 MHz Master Clock**
- **New 16-Pin Package for Higher Linecard Density**
- **AT&T D3/D4 and CCITT Compatible**
- **Variable Timing Mode for Flexible Digital Interface: Supports Data Rates from 64 KB to 2.048 MB**
- **Fully Differential Internal Architecture Enhances Noise Immunity**
- **Fixed Timing Mode for Standard 32-Channel Systems: 2.048 MHz Master Clock**
- **Low Power HMOS-E Technology**
  — 5 mW Typical Power Down
  — 140 mW Typical Operating
- **On Chip Auto Zero, Sample and Hold, and Precision Voltage References**
- **Compatible with Direct Mode Intel 2910A, 2911A, and 2912A Designs**

The Intel 2916 and 2917 are limited feature versions of Intel's 2913 and 2914 combination codec/filter chips. They are fully integrated PCM codecs with transmit/receive filters fabricated in a highly reliable and proven N-channel HMOS silicon gate technology (HMOS-E). These devices provide the functions that were formerly provided by two complex chips (2910A or 2911A and 2912A). Besides the higher level of integration, the performance of the 2916 and 2917 is superior to that of the separate devices.

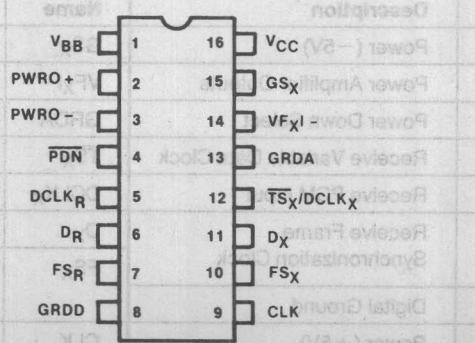
The primary applications for the 2916 and 2917 are in telephone systems:

- Switching—Digital PBX's and Central Office Switching Systems
- Subscriber Instruments—Digital Handsets and Office Workstations

Other possible applications can be found where the wide dynamic range (78 dB) and minimum conversion time (125 μs) are required for analog to digital interface functions:

- High Speed Modems
- Secure Communications
- Voice Store and Forward
- Digital Echo Cancellation

5



270156-1

Figure 1. Pin Configuration

September 1990
Order Number: 270156-002

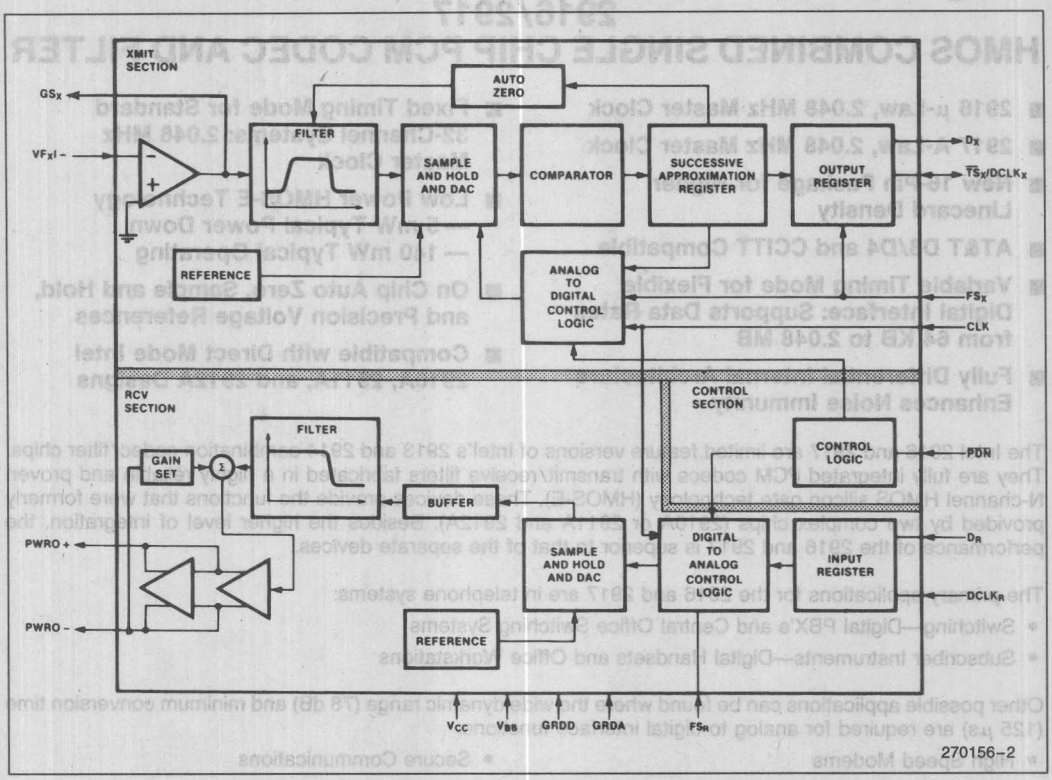


Figure 2. Block Diagram

Table 1. Pin Names

Name	Description	Name	Description
V_{BB}	Power (−5V)	GS_X	Transmit Gain Control
PWRO+, PWRO−	Power Amplifier Outputs	VF_XI-	Analog Input
$\overline{PDN}$	Power Down Select	GRDA	Analog Ground
$DCLK_R$	Receive Variable Data Clock	$\overline{TS_X}$	Timeslot Strobe/Buffer Enable
D_R	Receive PCM Input	$DCLK_X$	Transmit Variable Data Clock
FS_R	Receive Frame Synchronization Clock	D_X	Transmit PCM Output
		FS_X	Transmit Frame Synchronization Clock
GRDD	Digital Ground		
V_{CC}	Power (+5V)	CLK	Master Clock

Table 2. Pin Description

Symbol	Function
V_{BB}	Most negative supply, input voltage is −5 volts ±5%.
PWRO+	Non-inverting output of power amplifier. Can drive transformer hybrids or high impedance loads directly in either a differential or single ended configuration.
PWRO−	Inverting output of power amplifier. Functionally identical and complementary to PWRO+.
$\overline{PDN}$	Power down select. When $\overline{PDN}$ is TTL high, the device is active. When low, the device is powered down.
$DCLK_R$	Selects the fixed or variable data rate mode. When $DCLK_R$ is connected to V_{BB}, the fixed data rate mode is selected. When $DCLK_R$ is not connected to V_{BB}, the device operates in the variable data rate mode. In this mode $DCLK_R$ becomes the receive data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.
D_R	Receive PCM input. PCM data is clocked in on this lead on eight consecutive negative transitions of the receive data clock; CLK in the fixed data rate mode and $DCLK_R$ in variable data rate mode.
FS_R	8 KHz frame synchronization clock input/timeslot enable, receive channel. In variable data rate mode this signal must remain high for the entire length of the timeslot. The receive channel enters the standby state whenever FS_R is TTL low for 300 milliseconds.
GRDD	Digital ground for all internal logic circuits. Not internally tied to GRDA.
CLK	Master and data clock for the fixed data rate mode; master clock only in variable data rate mode.
FS_X	8 KHz frame synchronization clock input/timeslot enable, transmit channel. Operates independently but in an analogous manner to FS_R. The transmit channel enters the standby state whenever FS_X is TTL low for 300 milliseconds.
D_X	Transmit PCM output. PCM data is clocked out on this lead on eight consecutive positive transitions of the transmit data clock; CLK in fixed data rate mode and $DCLK_X$ in variable data rate mode.
$\overline{TS_X}/DCLK_X$	Transmit channel timeslot strobe (output) or data clock (input) for the transmit channel. In fixed data rate mode, this pin is an open drain output designed to be used as an enable signal for a three-state buffer. In variable data rate mode, this pin becomes the transmit data clock which operates at TTL levels from 64 Kb to 2.048 Mb data rates.
GRDA	Analog ground return for all internal voice circuits. Not internally connected to GRDD.
VF_XI-	Inverting analog input to uncommitted transmit operational amplifier.
GS_X	Output terminal of on-chip transmit channel input op amp. Internally, this is the voice signal input to the transmit filter.
V_{CC}	Most positive supply; input voltage is +5 volts ±5%.

5

FUNCTIONAL DESCRIPTION

The 2916 and 2917 provide the analog-to-digital and the digital-to-analog conversions and the transmit and receive filtering necessary to interface a full duplex (4 wires) voice telephone circuit with the PCM highways of a time division multiplexed (TDM) system. They are intended to be used at the analog termination of a PCM line.

The following major functions are provided:

- Bandpass filtering of the analog signals prior to encoding and after decoding
- Encoding and decoding of voice and call progress information
- Encoding and decoding of the signaling and supervision information

GENERAL OPERATION

System Reliability Features

The combochip can be powered up by pulsing FS_X and/or FS_R while a TTL high voltage is applied to $\overline{PDN}$, provided that all clocks and supplies are connected. The 2916 and 2917 have internal resets on power up (or when V_{BB} or V_{CC} are re-applied) in order to ensure validity of the digital outputs and thereby maintain integrity of the PCM highway.

On the transmit channel, digital outputs D_X and $\overline{TS}_X$ are held in a high impedance state for approximately four frames (500 μs) after power up or application of V_{BB} or V_{CC}. After this delay, D_X and $\overline{TS}_X$ will be functional and will occur in the proper timeslot. The analog circuits on the transmit side require approximately 60 milliseconds to reach their equilibrium value due to the autozero circuit settling time.

To enhance system reliability, $\overline{TS}_X$ and D_X will be placed in a high impedance state approximately 30 μs after an interruption of CLK.

Power Down and Standby Modes

To minimize power consumption, two power down modes are provided in which most 2916/2917 functions are disabled. Only the power down, clock, and frame sync buffers, which are required to power up the device, are enabled in these modes. As shown in Table 3, the digital outputs on the appropriate channels are placed in a high impedance state until the device returns to the active mode.

The Power Down mode utilizes an external control signal to the $\overline{PDN}$ pin. In this mode, power consumption is reduced to an average of 5 mW. The device is active when the signal is high and inactive when it is low. In the absence of any signal, the $\overline{PDN}$ pin floats to TTL high allowing the device to remain active continuously.

The Standby mode leaves the user an option of powering either channel down separately or powering the entire device down by selectively removing FS_X and/or FS_R. With both channels in the standby state, power consumption is reduced to an average of 12 mW. If transmit only operation is desired, FS_X should be applied to the device while FS_R is held low. Similarly, if receive only operation is desired, FS_R should be applied while FS_X is held low.

Fixed Data Rate Mode

Fixed data rate timing, which is 2910A and 2911A compatible, is selected by connecting $DCLK_R$ to V_{BB}. It employs master clock CLK, frame synchronization clocks FS_X and FS_R, and output $\overline{TS}_X$.

Table 3. Power-Down Methods

Device Status	Power-Down Method	Typical Power Consumption	Digital Output Status
Power Down Mode	$\overline{PDN}$ = TTL Low	5 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state within 10 μs.
Standby Mode	FS_X and FS_R are TTL Low	12 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state within 300 milliseconds.
Only Transmit is on Standby	FS_X is TTL Low	70 mW	$\overline{TS}_X$ and D_X are placed in a high impedance state within 300 milliseconds.
Only Receive is on Standby	FS_R is TTL Low	110 mW	

CLK serves as the master clock to operate the codec and filter sections and as the bit clock to clock the data in and out from the PCM highway. FS_X and FS_R are 8 KHz inputs which set the sampling frequency. $\overline{TS}_X$ is a timeslot strobe/buffer enable output which gates the PCM word onto the PCM highway when an external buffer is used to drive the line.

Data is transmitted on the highway at D_X on the first eight positive transitions of CLK following the rising edge of FS_X. Similarly, on the receive side, data is received on the first eight falling edges of CLK. The frequency of CLK must be 2.048 MHz. No other frequency of operation is allowed in the fixed data rate mode.

Variable Data Rate Mode

Variable data rate timing is selected by connecting $DCLK_R$ to the bit clock for the receive PCM highway rather than to V_{BB}. It employs master clock CLK, bit clocks $DCLK_R$ and $DCLK_X$, and frame synchronization clocks FS_R and FS_X.

Variable data rate timing allows for a flexible data frequency. It provides the ability to vary the frequency of the bit clocks, from 64 KHz to 2.048 MHz. The master clock is still restricted to 2.048 MHz.

In this mode, $DCLK_R$ and $DCLK_X$ become the data clocks for the receive and transmit PCM highways. While FS_X is high, PCM data from D_X is transmitted onto the highway on the next eight consecutive positive transitions of $DCLK_X$. Similarly, while FS_R is high, each PCM bit from the highway is received by D_R on the next eight consecutive negative transitions of $DCLK_R$.

On the transmit side, the PCM word will be repeated in all remaining timeslots in the 125 μs frame as long as $DCLK_X$ is pulsed and FS_X is held high. This feature allows the PCM word to be transmitted to the PCM highway more than once per frame, if desired, and is only available in the variable data rate mode.

Precision Voltage References

No external components are required with the combochip to provide the voltage reference function. Voltage references are generated on-chip and are calibrated during the manufacturing process. The technique uses a difference in sub-surface charge density between two suitably implanted MOS devices to derive a temperature and bias stable reference voltage. These references determine the gain and dynamic range characteristics of the device.

Separate references are supplied to the transmit and receive sections and each is trimmed independently during the manufacturing process. The reference value is then further trimmed in the gain setting op-amps to a final precision value. With this method the combochip can achieve the extremely accurate Digital Milliwatt Responses specified in the transmission parameters, providing the user a significant margin for error in other board components.

TRANSMIT OPERATION

Transmit Filter

The input section provides gain adjustment in the passband by means of an on-chip operational amplifier. This operational amplifier has a commoon mode range of ± 2.17 volts, a maximum DC offset of 25 mV, a minimum open loop voltage gain of 5000, and a unity gain bandwidth of typically 1 MHz. Gain of up to 20 dB can be set without degrading the performance of the filter. The load impedance to ground (GRDA) at the amplifier output (GS_X) must be greater than 10 kilohms in parallel with less than 50 pF. The input signal on lead VF_XI- can be either AC or DC coupled. The input op amp can only be used in the inverting mode as shown in Figure 3.

A low pass anti-aliasing section is included on-chip. This section typically provides 35 dB attenuation at the sampling frequency. No external components are required to provide the necessary anti-aliasing function for the switched capacitor section of the transmit filter.

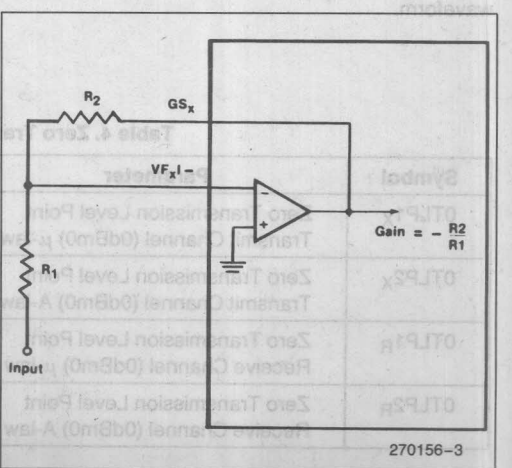


Figure 3. Transmit Filter Gain Adjustment

The passband section provides flatness and stopband attenuation which fulfills the AT&T D3/D4 channel bank transmission specification and CCITT recommendation G.714. The 2916 and 2917 specifications meet or exceed digital class 5 central office switching systems requirements. The transmit filter transfer characteristics and specifications will be within the limits shown in Figure 4.

A high pass section configuration was chosen to reject low frequency noise from 50 and 60 Hz power lines, 17 Hz European electric railroads, ringing frequencies and their harmonics, and other low frequency noise. Even though there is high rejection at these frequencies, the sharpness of the band edge gives low attenuation at 200 Hz. This feature allows the use of low-cost transformer hybrids without external components.

Encoding

The encoder internally samples the output of the transmit filter and holds each sample on an internal sample and hold capacitor. The encoder then performs an analog to digital conversion on a switched capacitor array. Digital data representing the sample is transmitted on the first eight data clock bits of the next frame.

An on-chip autozero circuit corrects for DC-offset on the input signal to the encoder. This autozero circuit uses the sign bit averaging technique; the sign bit from the encoder output is long term averaged and subtracted from the input to the encoder. In this way, all DC offset is removed from the encoder input waveform.

RECEIVE OPERATION

Decoding

The PCM word at the D_R lead is serially fetched on the first eight data clock bits of the frame. A D/A conversion is performed on the digital word and the corresponding analog sample is held on an internal sample and hold capacitor. This sample is then transferred to the receive filter.

Receive Filter

The receive filter provides passband flatness and stopband rejection which fulfills both the AT&T D3/D4 specification and CCITT recommendation G.714. The filter contains the required compensation for the (sin x)/x response of such decoders. The receive filter characteristics and specifications will be within the limits shown in Figure 5.

Receive Output Power Amplifiers

A balanced output amplifier is provided in order to allow maximum flexibility in output configuration. Either of the two outputs can be used single ended (referenced to GRDA) to drive single ended loads. Alternatively, the differential output will drive a bridged load directly. The output stage is capable of driving loads as low as 300 ohms single ended or 600 ohms differentially.

Transmission levels are specified relative to the receive channel output under digital milliwatt conditions, that is, when the digital input at D_R is the eight-code sequence specified in CCITT recommendation G.711.

Table 4. Zero Transmission Level Points

Symbol	Parameter	Value	Units	Test Conditions
$0TLP1_X$	Zero Transmission Level Point Transmit Channel (0dBm0) μ-law	+2.76 +1.00	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP2_X$	Zero Transmission Level Point Transmit Channel (0dBm0) A-law	+2.79 +1.03	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP1_R$	Zero Transmission Level Point Receive Channel (0dBm0) μ-law	+5.76 +4.00	dBm dBm	Referenced to 600Ω Referenced to 900Ω
$0TLP2_R$	Zero Transmission Level Point Receive Channel (0dBm0) A-law	+5.79 +4.03	dBm dBm	Referenced to 600Ω Referenced to 900Ω

ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias ..... −10°C to +80°C
Storage Temperature ..... −65°C to +150°C
V_{CC} and GRDD with Respect to V_{BB} ..... −0.3V to +15V
All Input and Output Voltages with Respect to V_{BB} ..... −0.3V to +15V
Power Dissipation 1.35W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

D.C. CHARACTERISTICS

(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{BB} = −5V ±5%, GRDA = 0V, GRDD = 0V, unless otherwise specified)

Typical values are for T_A = 25°C and nominal power supply values.

DIGITAL INTERFACE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{IL}	Low Level Input Current			10	μA	GRDD ≤ V_{IN} ≤ V_{IL} (Note 1)
I_{IH}	High Level Input Current			10	μA	V_{IH} ≤ V_{IN} ≤ V_{CC}
V_{IL}	Input Low Voltage			0.8	V	
V_{IH}	Input High Voltage	2.0			V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 3.2 mA at D_X, $\overline{TS}_X$
V_{OH}	Output High Voltage	2.4			V	I_{OH} = 9.6 mA at D_X
C_{OX}	Digital Output Capacitance(2)		5		pF	
C_{IN}	Digital Input Capacitance		5	10	pF	

POWER DISSIPATION

All measurements made at f_{DCLK} = 2.048 MHz, outputs unloaded.

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{CC1}	V_{CC} Operating Current(4)		14	19	mA	
I_{BB1}	V_{BB} Operating Current		−18	−24	mA	
I_{CC0}	V_{CC} Power Down Current		0.5	1.0	mA	$\overline{PDN}$ ≤ V_{IL}; after 10 μs
I_{BB0}	V_{BB} Power Down Current		−0.5	−1.0	mA	$\overline{PDN}$ ≤ V_{IL}; after 10 μs
I_{CCS}	V_{CC} Standby Current		1.2	2.4	mA	FS_X, FS_R ≤ V_{IL}; after 300 ms
I_{BBS}	V_{BB} Standby Current		−1.2	−2.4	mA	FS_X, FS_R ≤ V_{IL}; after 300 ms
P_{D1}	Operating Power Dissipation(3)		140	200	mW	
P_{D0}	Power Down Dissipation(3)		5	10	mW	$\overline{PDN}$ ≤ V_{IL}; after 10 μs
P_{ST}	Standby Power Dissipation(3)		12	25	mW	FS_X, FS_R ≤ V_{IL}

NOTES:
1. V_{IN} is the voltage on any digital pin.
2. Timing parameters are guaranteed based on a 100 pF load capacitance. Up to eight digital outputs may be connected to a common PCM highway without buffering, assuming a board capacitance of 60 pF.
3. With nominal power supply values.
4. V_{CC} applied last or simultaneously with V_{BB}.

5

ANALOG INTERFACE, TRANSMIT CHANNEL INPUT STAGE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{BX1}	Input Leakage Current, VF_XI-			100	nA	$-2.17V \leq V_{IN} \leq 2.17V$
R_{IXI}	Input Resistance, VF_XI-	10			MΩ	
V_{OSXI}	Input Offset Voltage, VF_XI-			25	mV	
A_{VOL}	DC Open Loop Voltage Gain, GS_X	5000				
f_C	Open Loop Unity Gain Bandwidth, GS_X		1		MHz	
C_{LXI}	Load Capacitance, GS_X			50	pF	
R_{LXI}	Minimum Load Resistance, GS_X	10			KΩ	

ANALOG INTERFACE, RECEIVE CHANNEL DRIVER AMPLIFIER STATE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
R_{ORA}	Output Resistance, PWRO+, PWRO−		1		Ω	
V_{OSRA}	Single-Ended Output DC Offset, PWRO+, PWRO−		75		mV	Relative to GRDA
C_{LRA}	Load Capacitance, PWRO+, PWRO−			100	pF	

A.C. CHARACTERISTICS—TRANSMISSION PARAMETERS

Unless otherwise noted, the analog input is a 0 dBm0, 1020 Hz sine wave.(1) Input amplifier is set for unity gain, inverting. The digital input is a PCM bit stream generated by passing a 0 dBm0, 1020 Hz sine wave through an ideal encoder. Receive output is measured single ended. All output levels are (sin x)/x corrected. Typical values are for $T_A = 25°C$ and nominal power supply values. ($T_A = 0°C$ to $+70°C$; $V_{CC} = +5V \pm 5\%$; $V_{BB} = -5V \pm 5\%$; GRDA = 0V; GRDD = 0V; unless otherwise specified).

GAIN AND DYNAMIC RANGE

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
EmW	Encoder Milliwatt Response (Transmit Gain Tolerance)	−0.18	±0.04	+0.18	dBm0	Signal Input of 1.064 Vrms μ-law Signal Input of 1.068 Vrms A-law $T_A = 25°C$, $V_{BB} = -5V$, $V_{CC} = +5V$
EmW_{TS}	EmW Variation with Temperature and Supplies	−0.07	±0.02	+0.07	dB	±5% Supplies, 0 to 70°C Relative to Nominal Conditions
DmW	Digital Milliwatt Response (Receive Gain Tolerance)	−0.18	±0.04	+0.18	dBm0	Measure Relative to $0TLP_R$. Signal Input per CCITT Recommendation G.711. Output Signal of 1000 Hz. $R_L = \infty$ $T_A = 25°C$; $V_{BB} = -5V$, $V_{CC} = +5V$.
DmW_{TS}	DmW Variation with Temperature and Supplies	−0.07	±0.02	+0.07	dB	±5% Supplies, 0 to 70°C

NOTE:
1. 0 dBm0 is defined as the zero reference point of the channel under test (0TLP). This corresponds to an analog signal input of 1.064 volts rms or an output of 1.503 volts rms (for μlaw).

GAIN TRACKING

Reference Level = −10 dBm0

Symbol	Parameter	2916 Min	2916 Max	2917 Min	2917 Max	Unit	Test Conditions
$GT1_X$	Transmit Gain Tracking Error Sinusoidal Input; μ-law		±0.25 ±0.5 ±1.2			dB dB dB	+3 to −40 dBm0 −40 to −50 dBm0 −50 to −55 dBm0
$GT2_X$	Transmit Gain Tracking Error Sinusoidal Input; A-law				±0.25 ±0.5 ±1.2	dB dB dB	+3 to −40 dBm0 −40 to −50 dBm0 −50 to −55 dBm0
$GT1_R$	Receive Gain Tracking Error Sinusoidal Input; μ-law		±0.25 ±0.5 ±1.2			dB dB dB	+3 to −40 dBm0 −40 to −50 dBm0 −50 to −55 dBm0 Measured at PWRO+, R_L = 300Ω
$GT2_R$	Receive Gain Tracking Error Sinusoidal Input; A-law				±0.25 ±0.5 ±1.2	dB dB dB	+3 to −40 dBm0 −40 to −50 dBm0 −50 to −55 dBm0 Measured at PWRO+, R_L = 300Ω

NOISE (All receive channel measurements are single ended)

Symbol	Parameter	2916 Min	2916 Typ	2916 Max	2917 Min	2917 Typ	2917 Max	Unit	Test Conditions
N_{XC1}	Transmit Noise, C-Message Weighted			15				dBrncO	Unity Gain
N_{XP}	Transmit Noise, Psophometrically Weighted						−75	dBm0p	Unity Gain
N_{RC1}	Receive Noise, C-Message Weighted: Quiet Code			11				dBrncO	D_R = 11111111
N_{RC2}	Receive Noise, C-Message Weighted: Sign Bit Toggle			12				dBrncO	Input to D_R is Zero Code with Sign Bit Toggle at 1 KHz Rate
N_{RP}	Receive Noise, Psophometrically Weighted						−79	dBm0p	D_R = Lowest Positive Decode Level
N_{SF}	Single Frequency Noise End to End Measurement			−50			−50	dBm0	CCITT G.712.4.2 Measure at PWRO+
$PSRR_1$	V_{CC} Power Supply Rejection, Transmit Channel		−30			−30		dB	Idle Channel; 200 mV P-P Signal on Supply; 0 to 50 KHz, Measure at D_X
$PSRR_2$	V_{BB} Power Supply Rejection, Transmit Channel		−30			−30		dB	Idle Channel; 200 mV P-P Signal on Supply; 0 to 50 KHz, Measure at D_X
$PSRR_3$	V_{CC} Power Supply Rejection, Receive Channel		−25			−25		dB	Idle Channel; 200 mV P-P Signal on Supply; Measure Narrow Band at PWRO+, 0 to 50 KHz
$PSRR_4$	V_{BB} Power Supply Rejection, Receive Channel		−25			−25		dB	Idle Channel; 200 mV P-P Signal on Supply; Measure Narrow Band at PWRO+, 0 to 50 KHz
CT_{TR}	Crosstalk, Transmit to Receive			−71			−71	dB	Input = 0 dBm0, Unity Gain, 1.02 KHz, D_R = Lowest Positive Decode Level, Measure at PWRO+
CT_{RT}	Crosstalk, Receive to Transmit			−71			−71	dB	D_R = 0 dBm0, 1.02 KHz, Measure at D_X

5

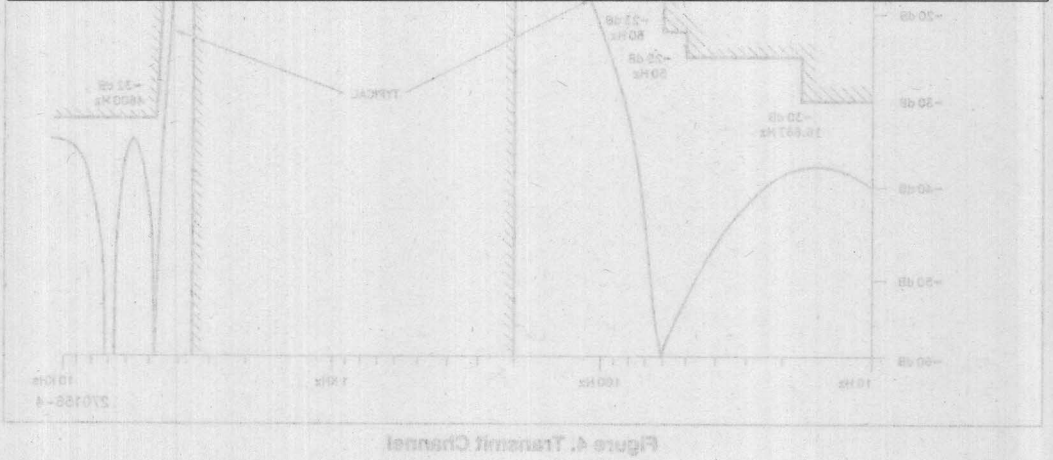
DISTORTION

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$SD1_X$	Transmit Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)	36 30 25			dB dB dB	0 dBm0 to −30 dBm0 −30 dBm0 to −40 dBm0 −40 dBm0 to −45 dBm0
$SD2_X$	Transmit Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)	36 30 25			dB dB dB	0 dBm0 to −30 dBm0 −30 dBm0 to −40 dBm0 −40 dBm0 to −45 dBm0
$SD1_R$	Receive Signal to Distortion, μ-Law Sinusoidal Input; CCITT G.714-Method 2 (2916)	36 30 25			dB dB dB	0 dBm0 to −30 dBm0 −30 dBm0 to −40 dBm0 −40 dBm0 to −45 dBm0
$SD2_R$	Receive Signal to Distortion, A-Law Sinusoidal Input; CCITT G.714-Method 2 (2917)	36 30 25			dB dB dB	0 dBm0 to −30 dBm0 −30 dBm0 to −40 dBm0 −40 dBm0 to −45 dBm0
DP_X	Transmit Single Frequency Distortion Products (2916)			−46	dBm0	AT&T Advisory #64 (3.8) 0 dBm0 Input Signal
DP_R	Receive Single Frequency Distortion Products (2916)			−46	dBm0	AT&T Advisory #64 (3.8) 0 dBm0 Input Signal
IMD_1	Intermodulation Distortion, End to End Measurement			−35	dB	CCITT G.712 (7.1)
IMD_2	Intermodulation Distortion, End to End Measurement			−49	dBm0	CCITT G.712 (7.2)
SOS	Spurious Out of Band Signals, End to End Measurement			−25	dBm0	CCITT G.712 (6.1)
SIS	Spurious In Band Signals, End to End Measurement			−40	dBm0	CCITT G.712 (9)
D_{AX}	Transmit Absolute Delay		245		μs	Fixed Data Rate. CLK_X = 2.048 MHz; 0 dBm0, 1.02 KHz Input Signal, Unity Gain. Measure at D_X.
D_{DX}	Transmit Differential Envelope Delay Relative to D_{AX}		170 95 45 105		μs μs μs μs	f = 500 Hz to 600 Hz f = 600 Hz to 1000 Hz f = 1000 Hz to 2600 Hz f = 2600 Hz to 2800 Hz
D_{AR}	Receive Absolute Delay		190		μs	Fixed Data Rate, CLK = 2.048 MHz; Digital Input is DMW Codes. Measure at PWRO+
D_{DR}	Receive Differential Envelope Delay Relative to D_{AR}		45 35 85 110		μs μs μs μs	f = 500 Hz to 600 Hz f = 600 Hz to 1000 Hz f = 1000 Hz to 2600 Hz f = 2600 Hz to 2800 Hz

TRANSMIT CHANNEL TRANSFER CHARACTERISTICS

Input amplifier is set for unity gain, inverting.

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
G_{RX}	Gain Relative to Gain at 1.02 KHz					0 dBm0 Signal Input at VF_XI-
	16.67 Hz			−30	dB	
	50 Hz			−25	dB	
	60 Hz			−23	dB	
	200 Hz	−1.8		−0.125	dB	
	300 to 3000 Hz	−0.125		+0.125	dB	
	3300 Hz	−0.35		+0.03	dB	
	3400 Hz	−0.7		−0.10	dB	
	4000 Hz			−14	dB	
	4600 Hz and Above			−32	dB	



5

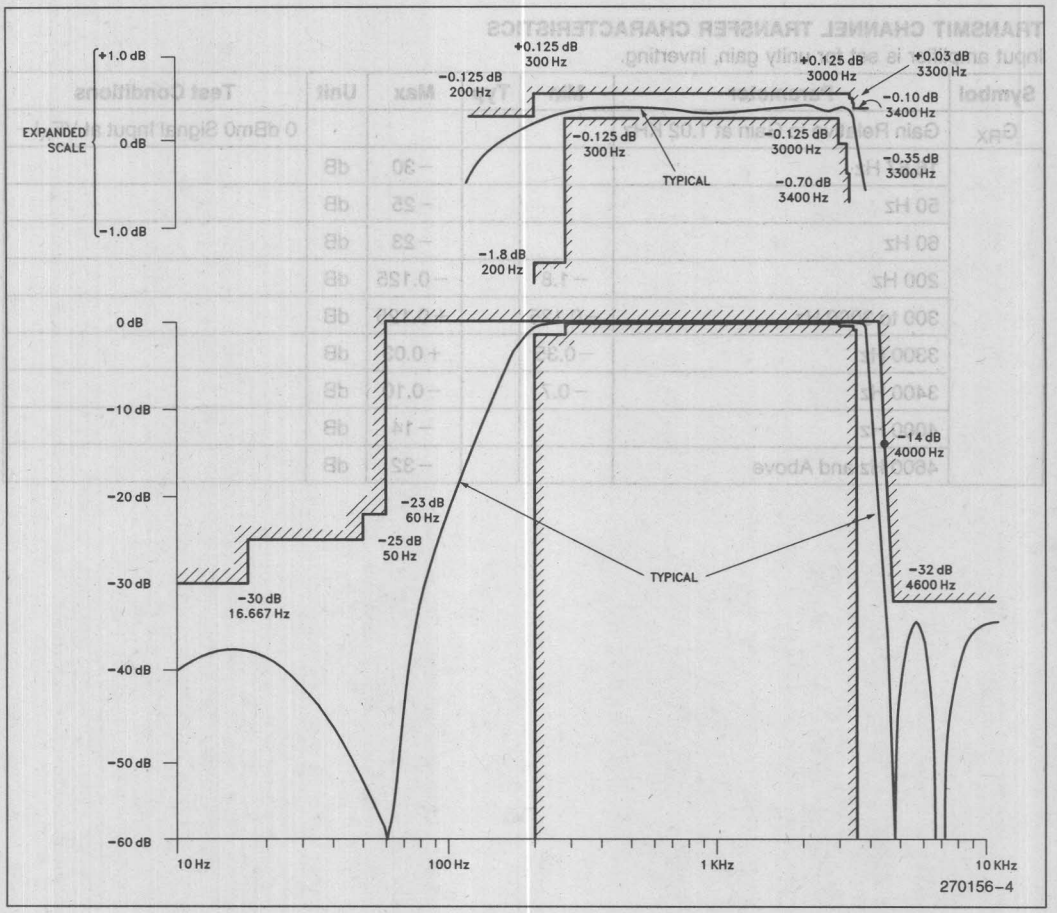
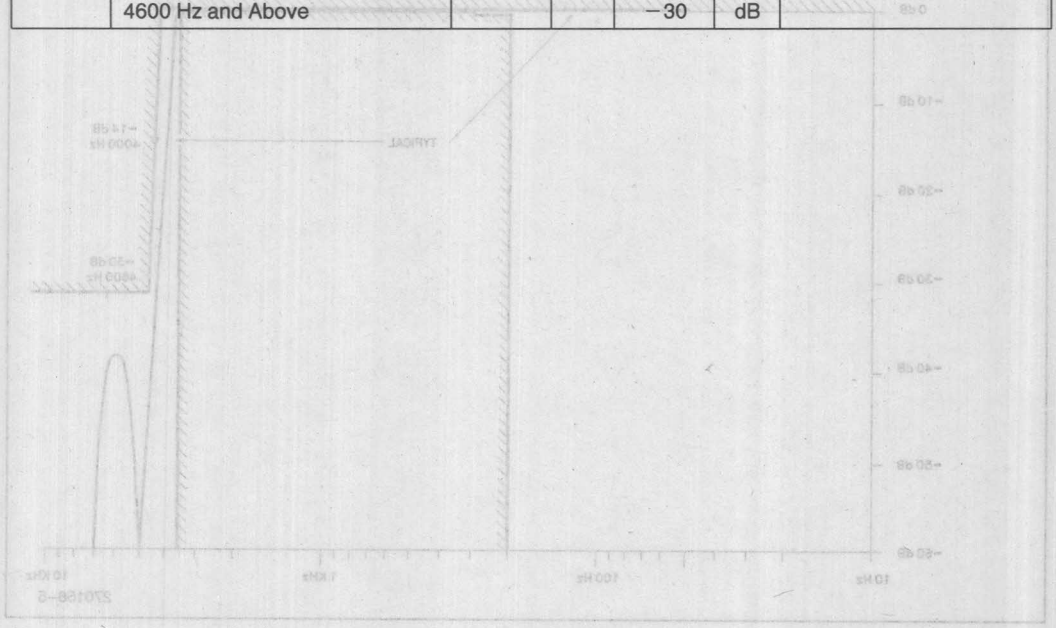


Figure 4. Transmit Channel

RECEIVE CHANNEL TRANSFER CHARACTERISTICS

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
G_{RR}	Gain Relative to Gain at 1.02 KHz					0 dBm0 Signal Input at D_R
	Below 200 Hz			+0.125	dB	
	200 Hz	−0.5		+0.125	dB	
	300 to 3000 Hz	+0.125		+0.125	dB	
	3300 Hz	−0.35		+0.03	dB	
	3400 Hz	−0.7		−0.1	dB	
	4000 Hz			−14	dB	
	4600 Hz and Above			−30	dB	



5

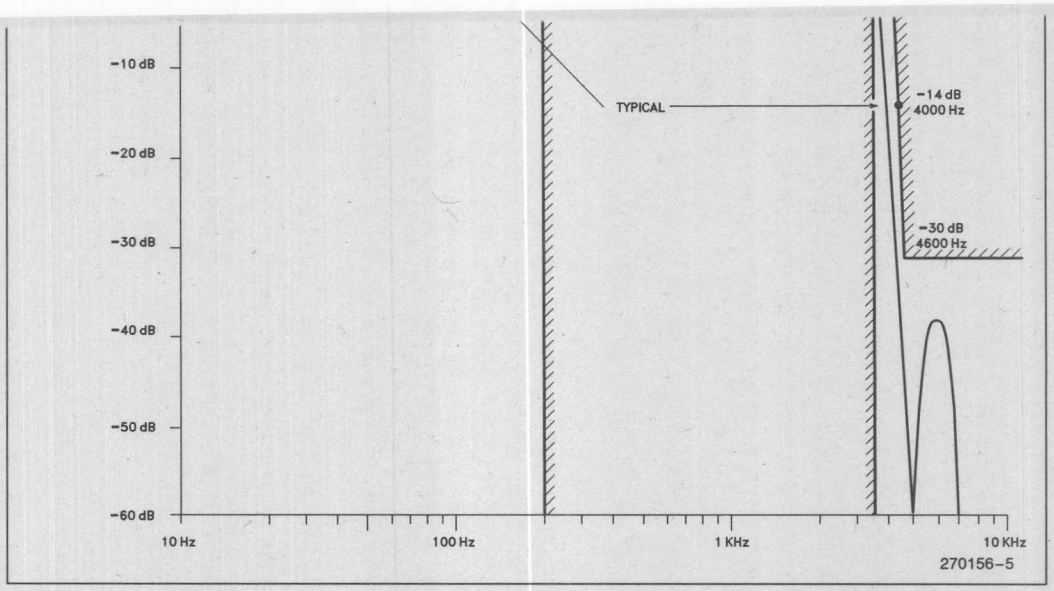


Figure 5. Receive Channel

A.C. CHARACTERISTICS—TIMING PARAMETERS

CLOCK SECTION

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{CY}	Clock Period, CLK	488			ns	f_{CLK} = 2.048 MHz
t_{CLK}	Clock Pulse Width, CLK	220			ns	
t_{DCLK}	Data Clock Pulse Width	220			ns	64 KHz ≤ f_{DCLK} ≤ 2.048 MHz
t_{CDC}	Clock Duty Cycle, CLK	45	50	55	%	
t_r, t_f	Clock Rise and Fall Time	5		30	ns	

TRANSMIT SECTION, FIXED DATA RATE MODE(1)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{DZX}	Data Enabled on TS Entry	0		145	ns	0 < C_{LOAD} < 100 pF
t_{DDX}	Data Delay from CLK	0		145	ns	0 < C_{LOAD} < 100 pF
t_{HZX}	Data Float on TS Exit	60		215	ns	C_{LOAD} = 0
t_{SON}	Timeslot X to Enable	0		145	ns	0 < C_{LOAD} < 100 pF
t_{SOFF}	Timeslot X to Disable	60		215	ns	C_{LOAD} = 0
t_{FSD}	Frame Sync Delay	100		t_{CLK}	ns	

RECEIVE SECTION, FIXED DATA RATE MODE

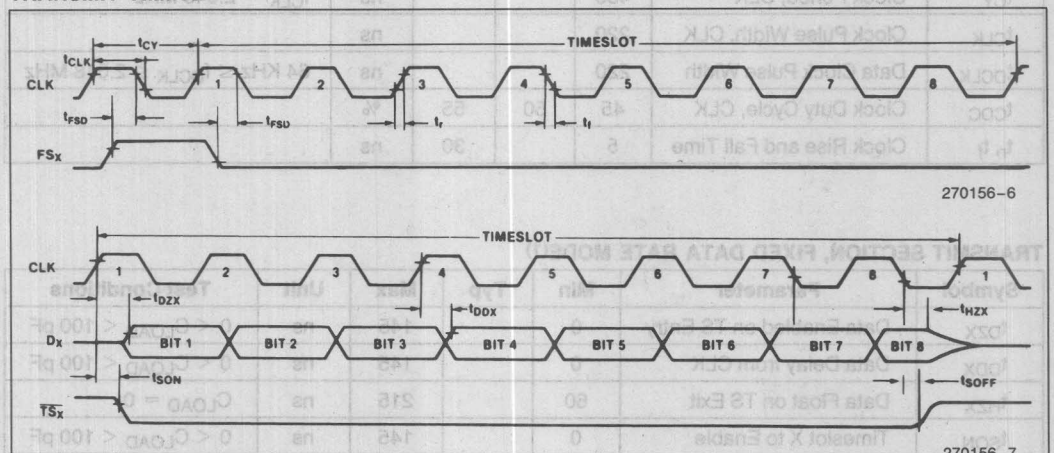
Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{DSR}	Receive Data Setup	10			ns	
t_{DHR}	Receive Data Hold	60			ns	
t_{FSD}	Frame Sync Delay	100		t_{CLK}	ns	

NOTE:
1. Timing parameters T_{DZX}, T_{HZX}, and T_{SOFF} are referenced to a high impedance state.

WAVEFORMS

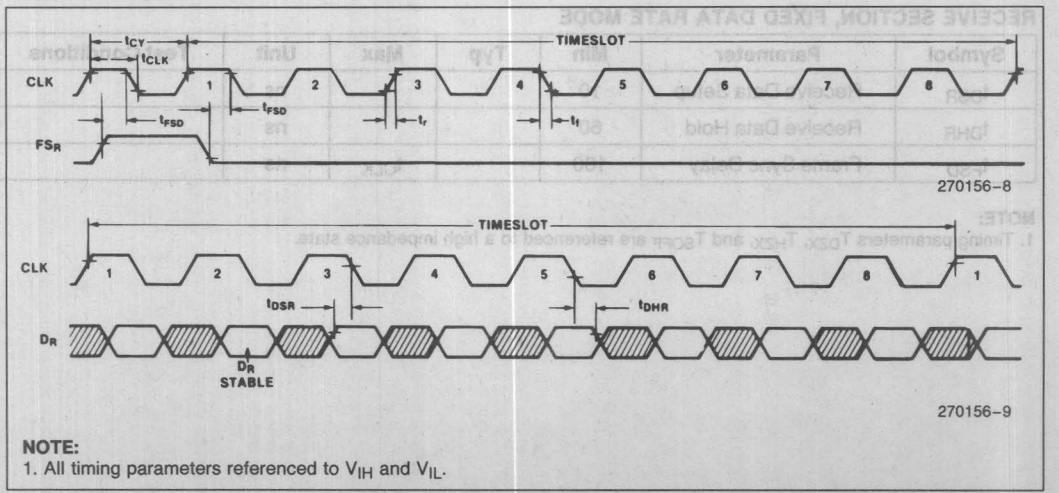
Fixed Data Rate Timing

TRANSMIT TIMING



NOTE:
1. All timing parameters referenced to V_{IH} and V_{IL} except t_{DZX}, t_{SOFF} and t_{HZX} which reference a high impedance state.

RECEIVE TIMING



NOTE:
1. All timing parameters referenced to V_{IH} and V_{IL}.

TRANSMIT SECTION, VARIABLE DATA RATE MODE(1)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{TSDX}	Timeslot Delay from $DCLK_X$(2)	140		$t_{DX}-140$	ns	
t_{FSD}	Frame Sync Delay	100		$t_{CY}-100$	ns	
t_{DDX}	Data Delay from $DCLK_X$	0		100	ns	$0 < C_{LOAD} < 100$ pF
t_{DON}	Timeslot to D_X Active	0		50	ns	$0 < C_{LOAD} < 100$ pF
t_{DOFF}	Timeslot to D_X Inactive	0		80	ns	$0 < C_{LOAD} < 100$ pF
t_{DX}	Data Clock Period	488		15620	ns	
t_{DFSX}	Data Delay from FS_X	0		140	ns	

RECEIVE SECTION, VARIABLE DATA RATE MODE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{TSDR}	Timeslot Delay from $DCLK_R$(3)	140		$t_{DR}-140$	ns	
t_{FSD}	Frame Sync Delay	100		$t_{CY}-100$	ns	
t_{DSR}	Data Setup Time	10			ns	
t_{DHR}	Data Hold Time	60			ns	
t_{DR}	Data Clock Period	488		15620	ns	
t_{SER}	Timeslot End Receive Time	60			ns	

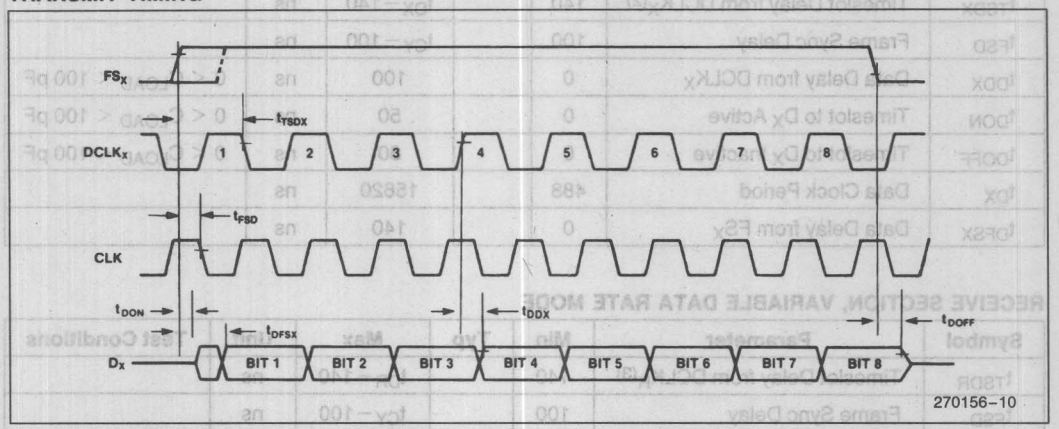
64 KB OPERATION, VARIABLE DATA RATE MODE

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
t_{FSLX}	Transmit Frame Sync Minimum Downtime	488			ns	FS_X is TTL High for Remainder of Frame
t_{FSLR}	Receive Frame Sync Minimum Downtime	1952			ns	FS_R is TTL High for Remainder of Frame
t_{DCLK}	Data Clock Pulse Width			10	μs	

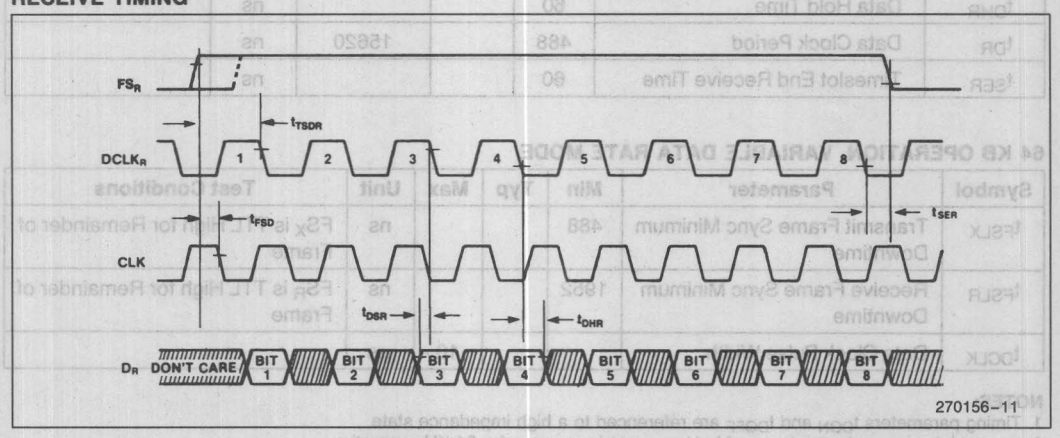
NOTES:
1. Timing parameters t_{DON} and t_{DOFF} are referenced to a high impedance state.
2. t_{FSLX} minimum requirements overrides t_{TSDX} maximum spec for 64 KHz operation.
3. t_{FSLR} minimum requirements overrides t_{TSDR} maximum spec for 64 KHz operation.

VARIABLE DATA RATE TIMING

TRANSMIT TIMING



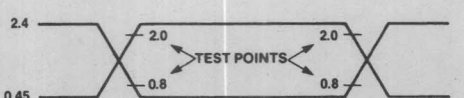
RECEIVE TIMING



NOTE:
1. All timing parameters referenced to V_{IH} and V_{IL} except t_{DON} and t_{OFF} which reference a high impedance state.

A.C. TESTING INPUT, OUTPUT WAVEFORM

INPUT/OUTPUT



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

APPLICATIONS INFORMATION
2910A/2911A/2912A

CODEC INTERFACE

The 2912A PCM Filter is designed to directly interface to the 2910A and 2911A Codecs as shown below. The transmit path is completed by connecting the VF_XO output of the 2912A to the coupling capacitor associated with the VF_X input of the 2910A and 2911A codecs. The receive path is completed by directly connecting the codec output VF_R to receive input of the 2912A VF_RI. The PDN input of the 2912A should be connected to the PDN output of the codec to allow the filter to be put in the power-down standby mode under control of the codec.

CLOCK INTERFACE

To assure proper operation, the CLK input of the 2912A should be connected to the same clock provided to receive bit clock, CLK_R of 2910A or 2911A Codec as shown below. The CLK0 input of the 2912A should be set to the proper voltage depending on the standard clock frequency chosen for the codec and filter.

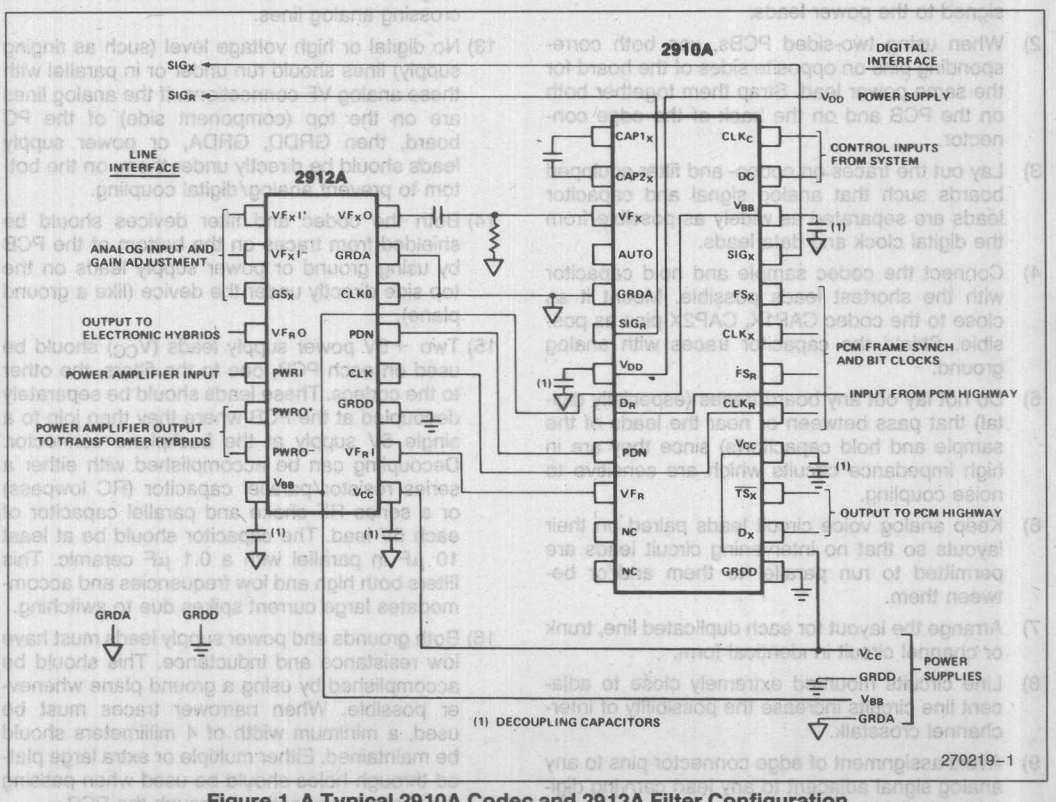


Figure 1. A Typical 2910A Codec and 2912A Filter Configuration

5

October 1986
Order Number: 270219-001

GROUNDING, DECOUPLING, AND LAYOUT RECOMMENDATIONS

The most important steps in designing a low noise line card are to insure that the layout of the circuit components and traces results in a minimum of cross coupling between analog and digital signals, and to provide well bypassed and clean power supplies, solid ground planes, and minimal lead lengths between components.

1) All power source leads should be bypassed to ground on each printed circuit board (PCB), on which codecs are provided. At least one electrolytic bypass capacitor (at least 50 μF) per board is recommended at the point where all power traces from the codecs and filters join prior to interfacing with the edge connector pins assigned to the power leads.
2) When using two-sided PCBs, use both corresponding pins on opposite sides of the board for the same power lead. Strap them together both on the PCB and on the back of the edge connector.
3) Lay out the traces on codec- and filter-equipped boards such that analog signal and capacitor leads are separated as widely as possible from the digital clock and data leads.
4) Connect the codec sample and hold capacitor with the shortest leads possible. Mount it as close to the codec CAP1X, CAP2X pins as possible. Shield the capacitor traces with analog ground.
5) Do not lay out any board traces (especially digital) that pass between or near the leads of the sample and hold capacitor(s) since they are in high impedance circuits which are sensitive to noise coupling.
6) Keep analog voice circuit leads paired on their layouts so that no intervening circuit leads are permitted to run parallel to them and/or between them.
7) Arrange the layout for each duplicated line, trunk or channel circuit in identical form.
8) Line circuits mounted extremely close to adjacent line circuits increase the possibility of interchannel crosstalk.
9) Avoid assignment of edge connector pins to any analog signal adjacent to any lead carrying digital (periodic) signals or power.
10) The optimum grounding configuration is to maintain separate digital and analog grounds on the circuit boards, and to carry these grounds back to the power supply with a low impedance connection. This keeps the grounds separate over the entire system except at the power supply.
11) The voltage difference between ground leads GRDA and GRDD (analog and digital ground) should not exceed two volts. One method of preventing any substantial voltage difference between leads GRDA and GRDD is to connect two diodes back to back in opposite directions across these two ground leads on each board.
12) Codec-filter pairs should be aligned so that pins 9 through 16 of the filter face pins 1 through 12 of the codec. This minimizes the distance for analog connections between devices and with no crossing analog lines.
13) No digital or high voltage level (such as ringing supply) lines should run under or in parallel with these analog VF connections. If the analog lines are on the top (component side) of the PC board, then GRDD, GRDA, or power supply leads should be directly under them, on the bottom to prevent analog/digital coupling.
14) Both the codec and filter devices should be shielded from traces on the bottom of the PCB by using ground or power supply leads on the top side directly under the device (like a ground plane).
15) Two +5V power supply leads (V_{CC}) should be used on each PCB, one to the filters, the other to the codecs. These leads should be separately decoupled at the PCB where they then join to a single 5V supply at the backplane connector. Decoupling can be accomplished with either a series resistor/parallel capacitor (RC lowpass) or a series RF choke and parallel capacitor of each 5V lead. The capacitor should be at least 10 μF in parallel with a 0.1 μF ceramic. This filters both high and low frequencies and accommodates large current spikes due to switching.
16) Both grounds and power supply leads must have low resistance and inductance. This should be accomplished by using a ground plane whenever possible. When narrower traces must be used, a minimum width of 4 millimeters should be maintained. Either multiple or extra large plated through holes should be used when passing the ground connections through the PCB.

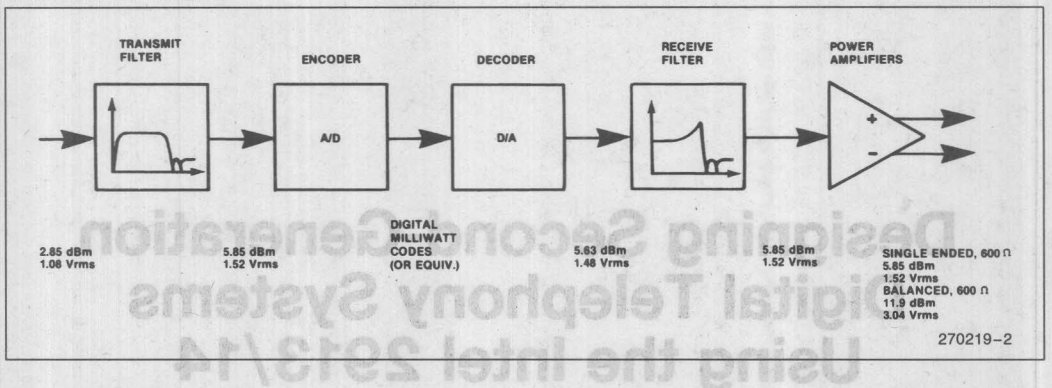
17) The 2912A PCM filter should have all power supplies bypassed to analog ground (GRDA). The 2910A/2911A Codec +5V power supplies should be bypassed to the digital ground (GRDD). This is appropriate when separate +5V power supply leads are used as suggested in item 15. The −5V and +12V supplies should be bypassed to analog ground (GRDA). Bypass capacitors at each device should be high frequency capacitors of approximately 0.1 to 1.0 μF value. Their lead lengths should be minimized by routing the capacitor leads to the appropriate ground plane under the device (either GRDA or GRDD).

18) Relay operation, ring voltage application, interruptions, and loop current surges can produce enormous transients. Leads carrying such signals must be routed well away from both analog and digital circuits on the line card and in backplanes. Lead pairs carrying current surges should be routed closely together to minimize possible inductive coupling. The microcomputer clock lead is particularly vulnerable, and should be buffered. Care should also be used in the backplane layout to prevent pickup surges. Any other latching components (relay buffers, etc.) should also be protected from surges.

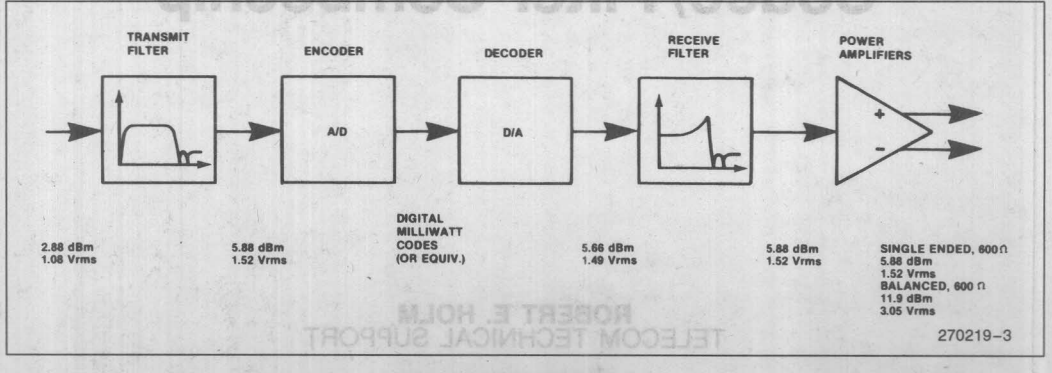
19) When not used, the AUTO pin should float with minimum PC board track area.

ZERO TRANSMISSION LEVEL POINTS

2910A/2912A 0 dBm0



2911A/2912A 0 dBm0



5

Designing Second-Generation Digital Telephony Systems Using the Intel 2913/14 Codec/Filter Combochip

ROBERT E. HOLM
TELECOM TECHNICAL SUPPORT

JOHN HUGGINS
TELECOM DESIGN ENGINEERING

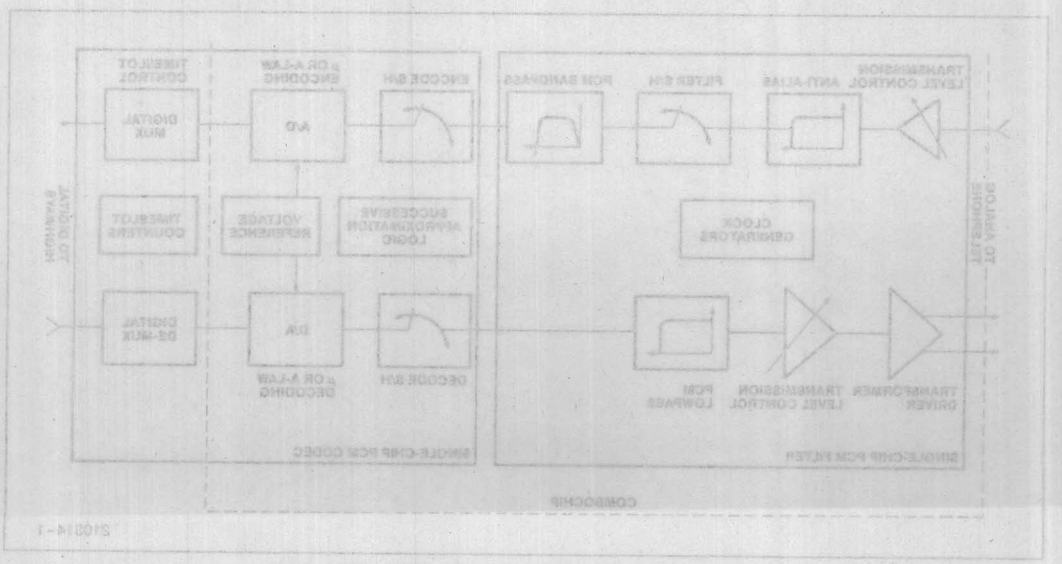
Order Number: 210314-002

DESIGNING SECOND-GENERATION DIGITAL TELEPHONY SYSTEMS USING THE INTEL 2913/14 CODEC/FILTER COMBOCHIP

CONTENTS

	PAGE
1.0 INTRODUCTION	5-90
1.1 Background	5-90
1.2 Comparison of First- and Second-Generation Component Capabilities	5-90
2.0 DESIGN CONSIDERATIONS	5-93
2.1 Operating and Test Mode Selection	5-94
2.2 Data Rate Modes	5-94
2.3 Margin to Performance Specifications	5-95
2.4 Power Conservation	5-97
2.5 Elimination of Gain Trim in the Line Circuit	5-99
2.6 Power Up/Down Considerations	5-100
3.0 OPERATING MODES	5-100
3.1 Fixed Data Rate Mode	5-100
3.2 Variable Data Rate Mode	5-100
3.3 On-Line Test Modes	5-102
4.0 MULTIMODE TEST CAPABILITIES	5-102
4.1 Design/Prototype Testing	5-102
4.2 Production Testing	5-104
4.3 Optional On-Line Testing	5-106

5



Note: See data sheet for latest specifications. Values given in this application note are for reference only, and were considered correct at the time of publication (Feb. 1982).

1.0 INTRODUCTION

This application note describes the features and capabilities of the 2913 and 2914 codec/filter combochips, and relates these capabilities to the design and manufacturing of transmission and switching linecards.

1.1 Background

The first generation of per line codecs (Intel 2910A/11A) and filters (Intel 2912A) economically integrated the analog-digital conversion circuits and PCM formatting circuits into one chip and the filtering and gain setting circuits into another chip. These two chips helped to make possible the rapid conversion to digital switching systems that has taken place in the last few years.

The second generation of Intel LSI PCM telephony components, the 2913/14 Combochip, extends the level of integration of the linecard by combining the codec and filter functions for each line on a single LSI chip. In the process of combining both functions, circuit design improvements have also improved performance, reduced external component count, lowered power dissipation, increased reliability, added new features, and maintained architectural transparency.

The 2913 and 2914 data sheet contains a complete description of both parts, including detailed discussions of each feature and specifications for timing and performance levels. This application note, in conjunction with the data sheet, describes in more detail how the new and improved features help in the design of second-generation linecards first by comparing the two generations of components to see where the improvements have been made, and then by discussing specific design considerations.

1.2 Comparison of First- and Second-Generation Component Capabilities

The combochip represents a higher level of component integration than the devices it replaces and, because of the economics of LSI (replacing two chips with one), ultimately will cost significantly less at the component level. But comparison of the combochip block diagram with first-generation single-chip codec and filter reveals few major functional differences. Figure 1 compares the first-generation codec and filter chips to the combochip. Both provide rigidly specified PCM capabilities of voice signal bandlimiting and nonlinear companded A/D and D/A conversion. The first on-chip reference voltage was introduced in the 2910/2911 single-chip codecs and is included in the combochip. The provision of uncommitted buffer amplifiers for flexible transmission level adjustment and enhanced analog output drive was a feature of the now standard 2912 switched-capacitor PCM filter is available on the combochip. Like-

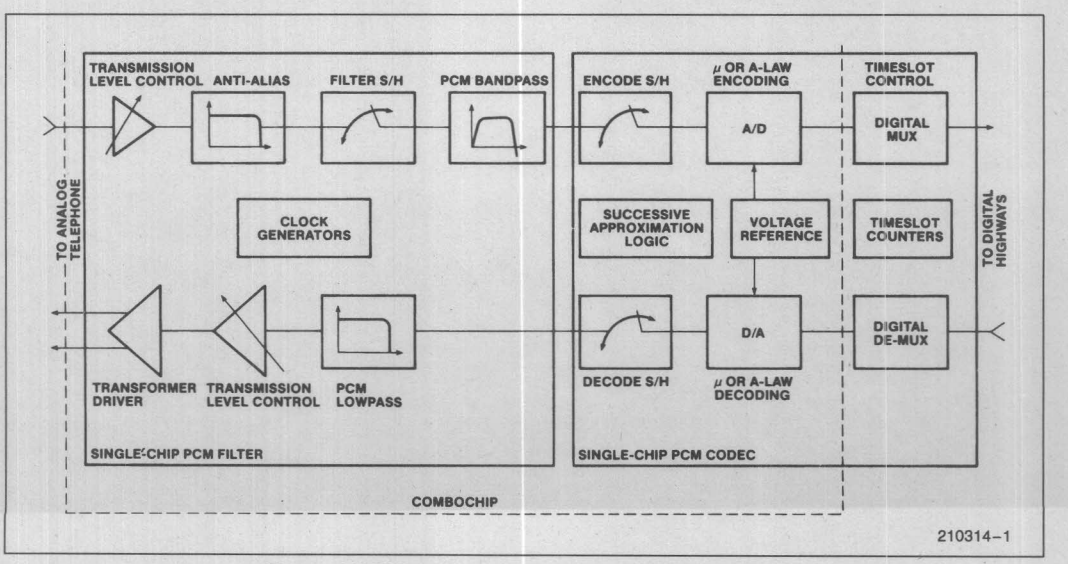


Figure 1. LSI Partitioning of Codec/Filter Functions

wise, independent transmit (A/D) and receive (D/A) analog voice channels which permit the two channels to be timed from independent (asynchronous) clock sources is common to the first- and second-generation devices. Finally, the ability to multiplex signalling bits on a bit-stealing basis from the digital side of the device has been duplicated on the combochip.

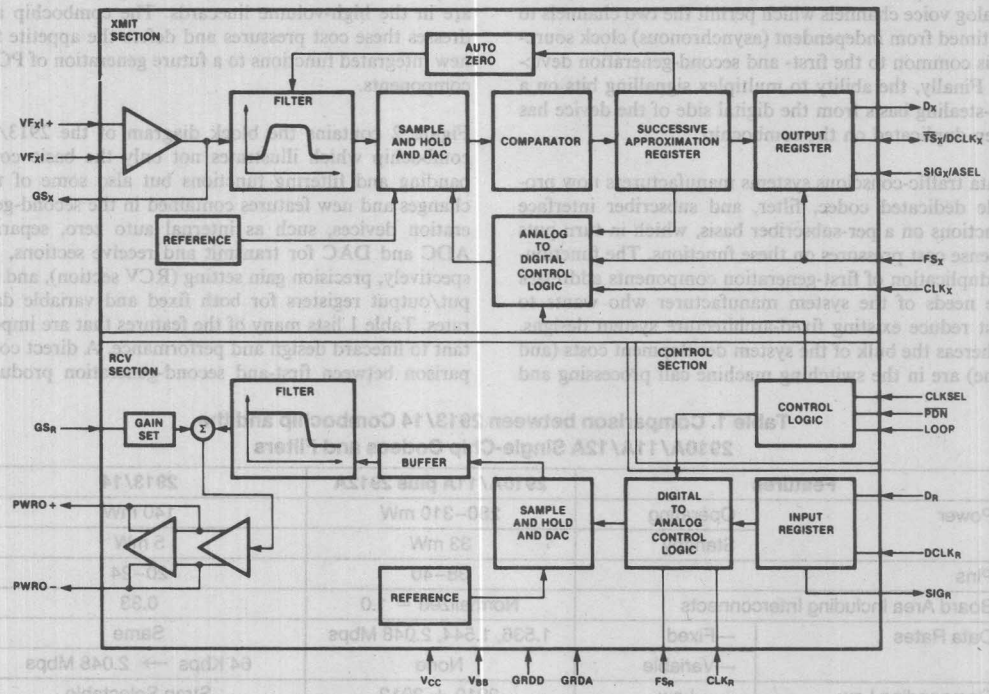
Data traffic-conscious systems manufacturers now provide dedicated codec, filter, and subscriber interface functions on a per-subscriber basis, which in turn puts intense cost pressures on these functions. The functional duplication of first-generation components addresses the needs of the system manufacturer who wants to cost reduce existing fixed-architecture system designs. Whereas the bulk of the system development costs (and time) are in the switching machine call processing and diagnostic software, the bulk of the production costs are in the high-volume linecards. The combochip addresses these cost pressures and defers the appetite for new integrated functions to a future generation of PCM components.

Figure 2 contains the block diagram of the 2913/14 combochip which illustrates not only the basic companding and filtering functions but also some of the changes and new features contained in the second-generation devices, such as internal auto zero, separate ADC and DAC for transmit and receive sections, respectively, precision gain setting (RCV section), and input/output registers for both fixed and variable data rates. Table 1 lists many of the features that are important to linecard design and performance. A direct comparison between first-and second-generation products

Table 1. Comparison between 2913/14 Combochip and the 2910A/11A/12A Single-Chip Codecs and Filters

Features		2910A/11A plus 2912A	2913/14
Power	Operating	280–310 mW	140 mW
	Standby	33 mW	5 mW
Pins		38–40	20–24
Board Area Including Interconnects		Normalized = 1.0	0.33
Data Rates	—Fixed	1.536, 1.544, 2.048 Mbps	Same
	—Variable	None	64 Kbps → 2.048 Mbps
Companding Law	—μ-Law	2910 + 2912	Strap Selectable
	—A-Law	2911 + 2912	
PSRR	1 KHz	30 dB	>35 dB
	> 10 KHz	Not Spec'd	> 35 dB
Gain Setting		Trim Using Pot Necessary	Precision Resistors Eliminate Trim Req.
Operating Modes	Direct	Yes	Yes
	Timeslot Assign	Yes	No
On-Chip V_{REF}		Yes	Yes
ICN — Half Channel Improvement		15 dBrnc0 Transmit 11 dBrnc0 Receive	15 dBrnc0 Transmit 11 dBrnc0 Receive
S/D — Half Channel Improvement		See Data Sheet	See Section 2.0
GT — Half Channel Improvement		See Data Sheet	See Section 2.0
Power Down (Standby)		PDN Pin	Frame Sync Removal or PDN Pin
Signalling		2910-8th Bit	2914-8th Bit
Auto Zero		External	Internal
S & H Caps		External Transmit Internal Receive	Internal
Test Modes		None	Design Tests Manufacturing Test On-Line Operational Tests
Encoder Implementation		Resistive Ladder	Capacitive Charge Redistribution Ladder
Filter/Gain Trim		Fuse Blowing ±0.2 dB	Fuse Blowing ±0.04 dB

5



(a) Combochip Block Diagram

V_{BB}	Power (−5V)	GS_X	Transmit Gain Control
PWRO+, PWRO−	Power Amplifier Outputs	VF_XI-, VF_XI+	Analog Inputs
GS_R	Receive Gain Control	GRDA	Analog Ground
$\overline{PDN}$	Power Down Select	NC	No Connect
CLKSEL	Master Clock Frequency Select	SIG_X	Transmit Signaling Input
LOOP	Analog Loop Back	ASEL	μ- or A-law Select
SIG_R	Receive Signaling Bit Output	$\overline{TS_X}$	Timeslot Strobe/Buffer Enable
$DCLK_R$	Receive Variable Data Clock	$DCLK_X$	Transmit Variable Data Clock
D_R	Receive PCM Input	D_X	Transmit PCM Output
FS_R	Receive Frame Synchronization Clock	FS_X	Transmit Frame Synchronization Clock
GRDD	Digital Ground	CLK_X	Transmit Master Clock
V_{CC}	Power (+5V)	CLK_R	Receive Master Clock

(b) Combochip Pin Names

Figure 2. Block Diagram of 2913/14 Combochip

shows the significant improvement in the combochip both in performance levels and system flexibility.

2.0 DESIGN CONSIDERATIONS

The key point with the 2913/14 is that it will result in a linecard that performs better and costs less than any two-chip codec/filter solution. The lower cost results from many factors, as seen in Table 2. Both direct replacement costs and less tangible design and manufacturing time savings combine to yield lower recurring and nonrecurring costs. As an example, the wider margins to transmission specs and the higher power supply rejection ratios of the 2913/14 will both shorten the design time needed to build and test the linecard prototype and reduce the reject rate on the manufacturing line.

Table 2. 2913/14 Factors which Lower the Cost of Linecard Design and Manufacturing

- Lower LSI Cost (2914 vs. 2910/11 + 2912)
- Fewer External Components
- Less Board Area
- Shorter Design/Prototype Cycle
- Better Yields/Higher Reliability
- Lower Power/Higher Density

Part of the recurring cost of linecard production is the efficiency of the manufacturing line in turning out each board. This is measured in both parts cost and time. Average manufacturing time is strongly effected by the line yield, i.e., the reject rate reliability. A linecard using the 2913/14 has many labor-saving features, which also increases the *reliability* of the manufacturing process. Some of these features are detailed in Table 3.

The combination of fewer parameters to trim (gain, reference voltage, etc.), tolerance to wider power supply variations, and on-chip test modes make the linecard very manufacturable compared to first-generation designs.

Probably the most obvious improvement in linecard design based around the 2913/14 is the reduction in linecard PCB area needed compared to two-chip designs. The combination of the codec and filter into a single package alone reduced the LSI area by one-third. Table 4 shows many of the other ways in which board area is conserved. In general, it reduces to fewer components, more on-chip features, and layout of the chip resulting in an efficient board layout which neatly separates the analog and digital signals both inside the chip and on the board.

Table 3. 2914 Factors which Increase Linecard Manufacturing Yields and Efficiency

- Higher Reliability
  - —Fewer connections and components
  - —More integrated packaging
  - —More margin to specs
  - —Lower power
  - —NMOS proven process
  - —Less sensitive to parameter variations
- Fewer Manufacturing Steps
  - —No gain trimming
  - —On chip V_{REF}
  - —Wide power supply tolerance
  - —On chip test modes
  - —Wide margins to spec

Table 4. Design Factors for 2914 which Reduce Linecard PCB Area

- Integrated Packaging
  - —2914 vs. 2910/11 + 2912 = 1/3 board area
  - —2913 takes even less space
- Fewer Interconnects/Components
  - —Codec/filter combined
  - —On-chip reference voltage
  - —On-chip auto zero
  - —On-chip capacitors
  - —No gain trim components
  - —No voltage regulators
- Efficient Layout (Facilitates Auto Insertion)
  - —Analog/digital sections separated on chip
  - —Digital traces can cross under chip
  - —Two power supplies only
  - —Low power/high density

5

Table 5. 2913/14 Operating Mode Options Add Flexibility to Linecard Design

Option	Mode Control Pins	Results of Mode Selection	
		2914 (24 Pin)	2913 (20 Pin)
Companding Law	SIGX/ASEL	A-Law or μ-Law + Signalling	A-Law/μ-Law, no Signalling
Power Down	$\overline{PDN}$	Transmit & Receive Side Go To Standby Power (5 mW)	
	FS_X & FS_R Removed	Same (12 mW)	
	FS_X Removed	Transmit Side Goes to Standby (110 mW)	
	FS_R Removed	Receive Side Goes to Standby (70 mW)	
Data Rate	= V_{CC}/GRDD/V_{BB} $DCLK_R$ = V_{BB}	1.536/1.544/2.048 Mbps in Fixed Data Rate Mode	
	= V_{CC}/GRDD/V_{BB} $DCLK_B$ = Clock	Variable Data Rate Mode from 64 Kbps to 2.048 Mbps, No Signalling	
Test Modes	LOOP = V_{CC}	Implements Analog Loopback	No Loopback Capability
	$\overline{PDN}$ = V_{BB}	Provides Access to Transmit Codec Through ASEL and $\overline{TSX}$ Pins	
	D_R = V_{BB}	Provides Access to RCV Filter Input at $DCLK_R$ and Transmit Filter Outputs at ASEL and $\overline{TSX}$ Pins	

Many of the factors discussed—which result in efficient, cost-effective linecard designs—are discussed in more detail both in the 2913/14 data sheet and in the following sections of this note.

2.1 Operating and Test Mode Selection

A key to designing with the 2913/14 combo is the wide range of options available in configuring, either with strap options or in real time, the different modes of operation. The 2913 combochip (20 pins) is specifically aimed at synchronous switching systems (remote concentrators, PABXs, central offices) where small package size is especially desirable. The 2914 combochip (24 pins) has additional features which are most suitable for applications requiring 8th-bit signalling, asynchronous operation, and remote testing of transmission paths (e.g., channel banks). Once the specific device is selected, there is a wide range of operating modes to use in the card design, as seen in Table 5. This table lists the optional parameters and the pins which control the operating mode. The result of selecting a mode is listed for both the 2913 and 2914.

The purpose of offering these options is to ensure that the 2913/14 combo will accommodate any existing linecard design with architectural transparency. At the same time, features were designed in to facilitate design and manufacturing testing to reduce overall cost of development and production.

2.2 Data Rate Modes

Any rapid conversion scenario presumes that the combochip will fit existing system architectures (retrofit) without significant system timing, control, or software modifications. To this end, two distinct user-selectable timing modes are possible with the combochip. For purposes of discussion, these are designated (a) fixed data rate timing (FDRT) and (b) variable data rate timing (VDRT).

FDRT is identical to the 2910/2911 codec timing in which a single high-speed clock serves both as master clock for the codec/filter internal conversion/filtering functions and as PCM bit clock for the high-speed serial PCM data bus over which the combochip transmits and receives its digitized voice code words. In this mode, PCM bit rates are necessarily confined to one of three distinct frequencies (1.536 MHz, 1.544 MHz, or 2.048 MHz). Many recently designed systems employ this type of timing which is sometimes referred to as burst-mode timing because of the low duty cycle of each timeslot (i.e., channel) on the time division multiplexed PCM bus. It is possible for up to 32 active combochips to share the same serial PCM bus with FDRT.

VDRT (sometimes referred to as shift register timing), by comparison, utilizes one high-speed master clock for the combochip internal conversion/filtering functions and a separate, variable frequency, clock as the PCM bit clock for the serial PCM data bus. Because the serial PCM data rate is independent of internal conversion timing, there is considerable flexibility in the choice of PCM data rate. In this mode the master clock is permitted to be 1.536 MHz, 1.544 MHz, or 2.048 MHz, while the bit clock can be any rate between 64 KHz and 2.048 MHz. In this mode it is possible to have a dedicated serial bus for each combochip or to share a single serial PCM bus among as many as 32 active combochips.

Thus, the two predominant timing configurations of present system architectures are served by the same device, allowing, in many cases, linecard redesign without modification of any common system hardware or software. Additional details relating to the design of systems using either mode are found in section 3.0.

2.3 Margin to Performance Specifications

The combochip benefits from design, manufacturing, and test experience with first-generation PCM products on the part of the system manufacturer, component suppliers, and test equipment suppliers. The sub-millivolt PCM measurement levels and tens of microvolts accuracy requirements on the lowest signal measurements often result in tester correlation problems, yield losses, and excess costs for system and PCM component manufacturers alike. Thus additional performance margin built into the PCM components themselves will have its effect on line circuit costs even though the system transmission specifications may not reflect the improved performance margin.

Half channel measurements have been made of the transmission parameters—gain tracking (GT), signal to distortion ratio (S/D), and idle channel noise (ICN).

Gain Tracking—Figure 3 shows the gain tracking data for both the transmit and receive sides of the combo using both sine wave testing (CCITT G712.11 Method 2) and white noise testing (CCITT G712.11 Method 1). The data shows a performance very nearly equal to the theoretically best achievable using both test techniques. End to end measurements, although not spec'd, also show a corresponding good performance with errors less than or equal to the sum of the half channel values.

Signal to Distortion Ratio—This is a measure of the system linearity and the accuracy in implementing the companding codes. Figure 4 shows the excellent perfor-

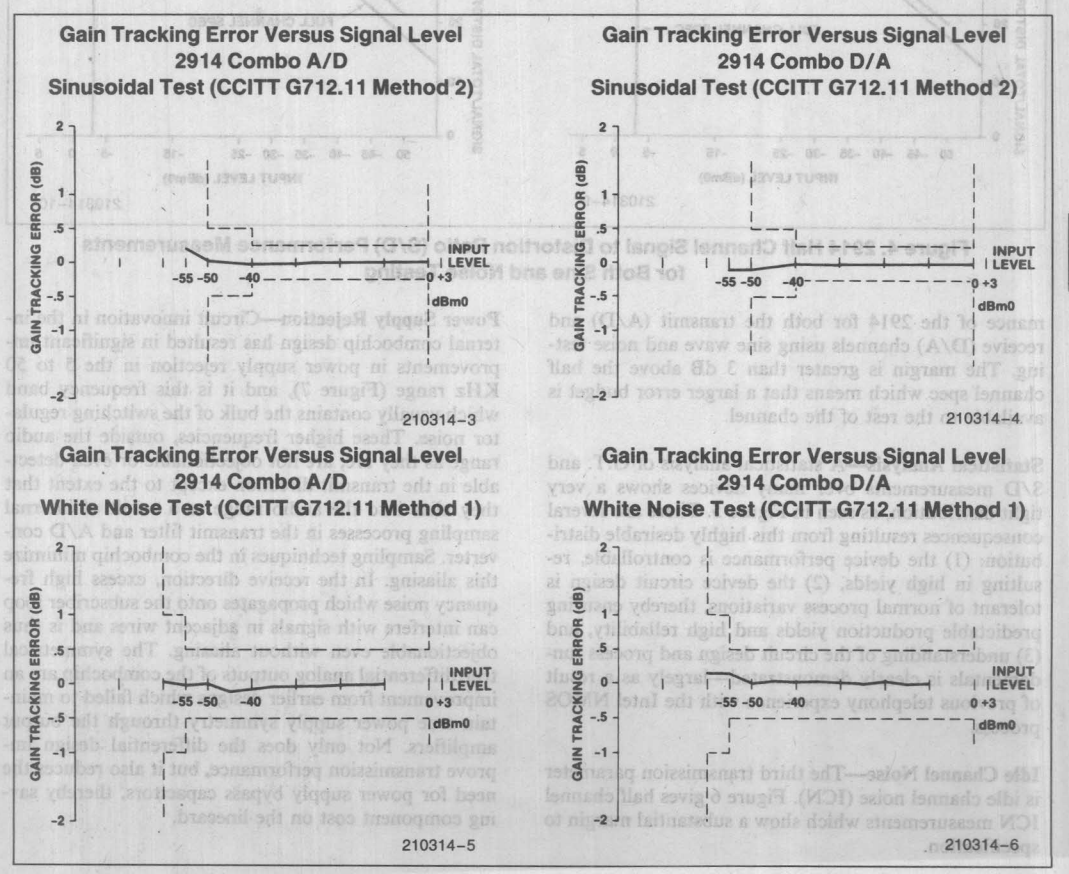


Figure 3. 2914 Half Channel Gain Tracking Performance Measurements for Both Sine and Noise Testing

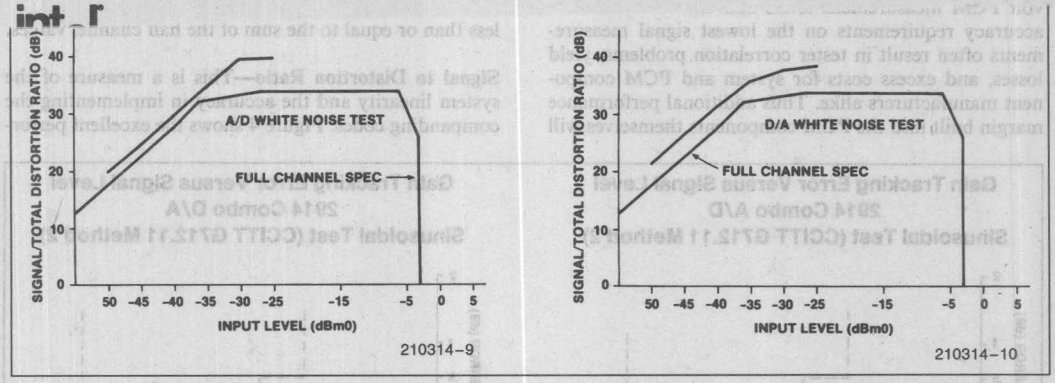


Figure 4. 2914 Half Channel Signal to Distortion Ratio (S/D) Performance Measurements for Both Sine and Noise Testing

mance of the 2914 for both the transmit (A/D) and receive (D/A) channels using sine wave and noise testing. The margin is greater than 3 dB above the half channel spec which means that a larger error budget is available to the rest of the channel.

Statistical Analysis—A statistical analysis of G.T. and S/D measurements over many devices shows a very tight distribution, as seen in Figure 5. There are several consequences resulting from this highly desirable distribution: (1) the device performance is controllable, resulting in high yields, (2) the device circuit design is tolerant of normal process variations, thereby ensuring predictable production yields and high reliability, and (3) understanding of the circuit design and process fundamentals is clearly demonstrated—largely as a result of previous telephony experience with the Intel NMOS process.

Idle Channel Noise—The third transmission parameter is idle channel noise (ICN). Figure 6 gives half channel ICN measurements which show a substantial margin to specification.

Power Supply Rejection—Circuit innovation in the internal combochip design has resulted in significant improvements in power supply rejection in the 5 to 50 KHz range (Figure 7), and it is this frequency band which usually contains the bulk of the switching regulator noise. These higher frequencies, outside the audio range as they are, are not objectionable or even detectable in the transmit direction except to the extent that they alias into the audio range as a result of internal sampling processes in the transmit filter and A/D converter. Sampling techniques in the combochip minimize this aliasing. In the receive direction, excess high frequency noise which propagates onto the subscriber loop can interfere with signals in adjacent wires and is thus objectionable even without aliasing. The symmetrical true differential analog outputs of the combochip are an improvement from earlier designs which failed to maintain true power supply symmetry through the output amplifiers. Not only does the differential design improve transmission performance, but it also reduces the need for power supply bypass capacitors, thereby saving component cost on the linecard.

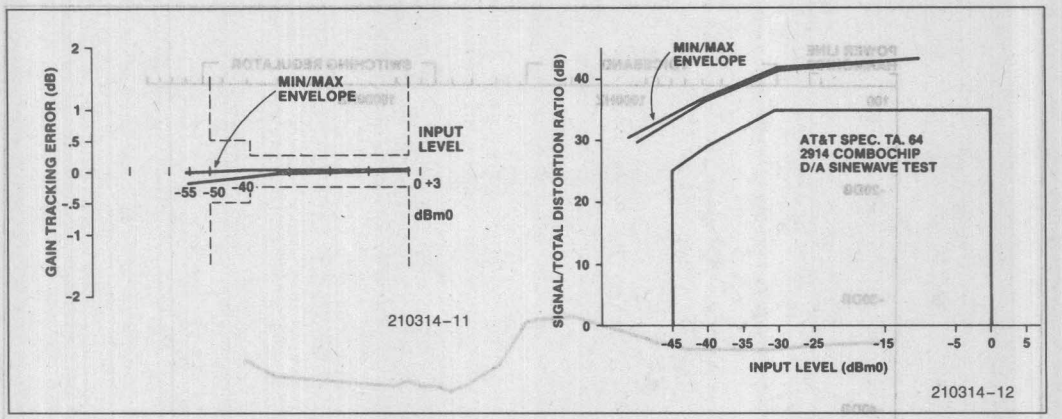


Figure 5. Statistical Analysis of Transmission Performance Showing Tight Distribution Over Many Devices

	Weighting	ICN
A/D	C Message	15 dBrnC0
D/A	C Message	11 dBrnC0

Figure 6. 2914 Idle Channel Noise (ICN) Measurements

Autozero—The autozero circuit is contained completely on-chip. It automatically centers the signal/noise distribution at the encoder input. This ensures minimal ICN due to bit toggling and also maintains maximum sensitivity to the AC signals of interest.

2.4 Power Conservation

Figure 8 illustrates typical power consumption and office equipment dissipation for a resistive line biasing arrangement (with no loop current limiting) and for the per-line PCM components. It can be seen that overall line circuit power consumption and dissipation are strong functions of subscriber loop resistance, and are dominated by line biasing current regardless of loop length. It can also be seen that the combochip achieves significant reductions in PCM component contributions relative to both the 2910A/2912A and 2910/2912. Present residential traffic characteristics are such that the PCM components are active less than 10% of the time, and in its low-power standby state, the combochip power dissipation drops to typically 5 mW as the line current (and dissipation) goes to its background on-hook leakage level of typically a few milliwatts (but for very leaky lines, as much as 50 mW–500 mW).

The concern for linecard power consumption and dissipation is related both to the cost of providing power and to the system density problem involving convection heat removal from the linecards. Consequently, much recent line circuit development activity centers on elimination of the inefficient resistive line current feed both by current limiting in short loops and by more exotic and expensive per-line dc-dc converters. For both present-generation designs and cost-reduction redesigns, the typical combochip dissipation of 140 mW active/5 mW standby will allow system board packing density improvements and power supply cost reductions.

A closer look at the effect of loading (duty cycle) on the average power dissipation of a combochip is given in Table 6. Typical loading percents run as low as 5% for very large switching systems (thousands of lines) up to 100% in nonswitching applications such as channel banks. Clearly, the average power dissipation in a typical switching system is below 35 mW which facilitates board packing density and cost of power considerations.

Table 6. Typical Power Dissipation Per Line Using 2914 Combochip

	Duty Cycle	Power Dissipation
Central Office	5%	12 mW
PABX	15%	25 mW
Peak Hour C.O.	50%	73 mW
Channel Bank	100%	140 mW

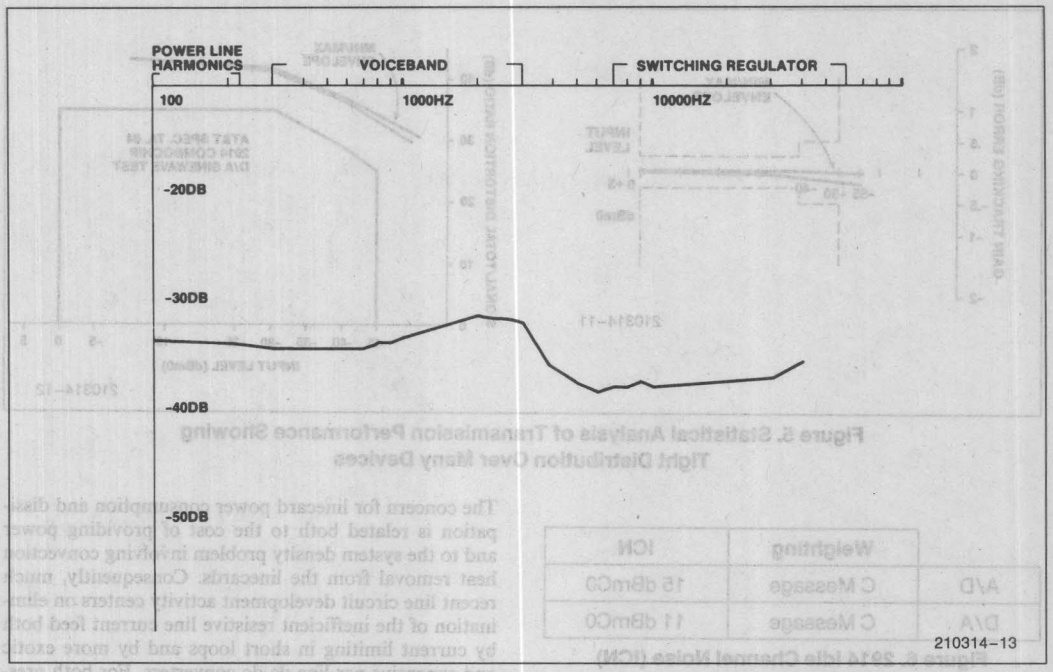


Figure 7. Wideband 2914 Power Supply Rejection Ratio (PSRR)

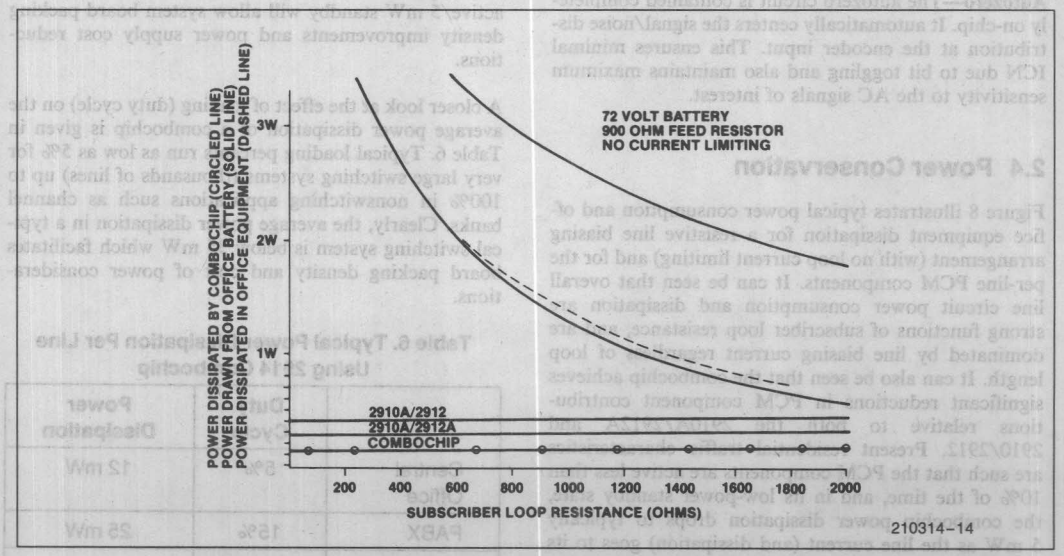


Figure 8. Line Circuit Power Consumption and Dissipation Curves

2.5 Elimination of Gain Trim in the Line Circuit

Four resistors—R1–R4 of Figure 9—on the transformer side of the PCM components are used to establish appropriate transmission levels at the PCM components and are, at first glance, equivalent in the two cases. However, a significant reduction in linecard manufacturing costs associated with individual line trim (or mop-up) is possible with the combochip. The need for this trim is dictated by system gain contrast specifications which typically require that the line-to-line gain variation shall not exceed 0.5 dB, which translates to 0.25 dB for each (transmit and receive) channel. Table 7 shows that the major portion of this gain variation has previously been in the nominal insertion loss of the PCM filter and in the uncertainty of the reference voltage of the codec. With this cumulative 0.15 dB uncertainty in the PCM components themselves, the system manufacturer had no choice but to resort to the cost and manufacturing complexity of the active trim. The combochip, however, can be trimmed during its manufacture to a nominal tolerance of ±0.04 dB which includes uncertainties in both the filter and codec voltage reference functions. This leaves 0.21 dB uncertainty to variations in the other line circuit elements and to temperature and supply variations.

The variation in combochip gain with supply and temperature has also been improved to allow as low as

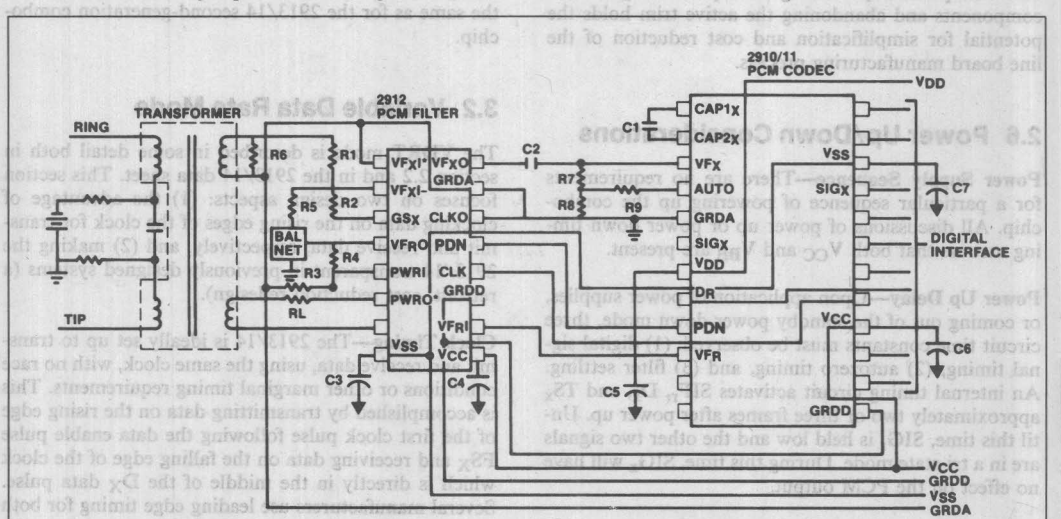


Figure 9. Schematics of the Codec/Filter Function and the 2/4 Wire Hybrid Transformers

Table 7. Gain Trim Budget for Codec/Filter Functions

Device	Manufacturing Uncertainty (Initial)	ΔT ΔSupplies	Total	Variation* Budget for Other Components
2910	±0.1	±0.1		
2912	±0.05	±0.05		
	±0.15	±0.15	±0.3 dB	0 dB
2914	±0.04	±0.08	±0.12 dB	±0.13 dB

*Assumes 0.5 dB end to end gain contrast specifications.

0.08 dB variation over supplies and temperature so that more than half the system specification could be reserved for transformer, wiring, and resistor uncertainties. This possibility of using fixed precision gain trim components and abandoning the active trim holds the potential for simplification and cost reduction of the line board manufacturing process.

2.6 Power Up/Down Considerations

Power Supply Sequence—There are no requirements for a particular sequence of powering up the combochip. All discussions of power up or power down timing assume that both V_{CC} and V_{BB} are present.

Power Up Delay—Upon application of power supplies, or coming out of the standby power down mode, three circuit time constants must be observed: (1) digital signal timing, (2) autozero timing, and (3) filter settling. An internal timing circuit activates SIF_r, D_x, and TS_x approximately two or three frames after power up. Until this time, SIG_r is held low and the other two signals are in a tri-state mode. During this time, SIG_x will have no effect on the PCM output.

Power Down Modes—These modes are described in detail in Table 3 of the 2913/14 data sheet except for a fail-safe mode in case CLK_x is interrupted. If this should happen, both D_x and TS_x go into the tri-state mode until the clock is restored. This ensures the safety of the PCM highway should the interrupted clock be a local problem.

3.0 OPERATING MODES

There are three basic operating modes that are supported by the 2913/14: fixed data rate timing (FDRT), variable data rate timing (VDRT), and on-line testing.

3.1 Fixed Data Rate Mode

The FDRT mode is described in some detail in both section 2.2 of this note and in the 2913/14 data sheet. In addition, Intel Application Note AP-64 (Data Conversion, Switching, and Transmission using the Intel 2910A/2911A codec and 2912 PCM filter) also describes the basics of using the fixed data rate mode for first-generation codecs and filters which is essentially the same as for the 2913/14 second-generation combochip.

3.2 Variable Data Rate Mode

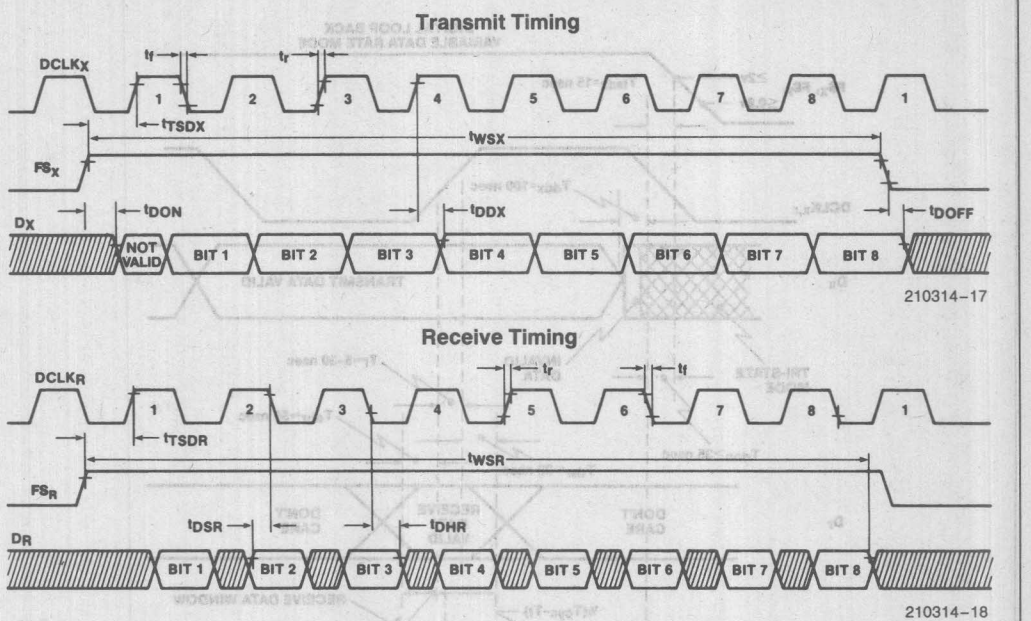
The VDRT mode is described in some detail both in section 2.2 and in the 2913/14 data sheet. This section focuses on two design aspects: (1) the advantage of clocking data on the rising edges of the clock for transmit and receive data, respectively, and (2) making the 2913/14 transparent in previously designed systems (a retrofit, cost reduction redesign).

Clock Timing—The 2913/14 is ideally set up to transmit and receive data, using the same clock, with no race conditions or other marginal timing requirements. This is accomplished by transmitting data on the rising edge of the first clock pulse following the data enable pulse FS_X and receiving data on the falling edge of the clock which is directly in the middle of the D_X data pulse. Several manufacturers use leading edge timing for both transmit and receive requiring an inversion of the receive clock.

Figure 10 shows the transmit and receive clock and data timing for an entire time slot of data. A closer look at the timing functions is given in Figure 11 which looks specifically at the first clock cycle after the transmit data enable FS_X.

According to the 2913/14 data sheet, the frame sync/data enable FS_X must precede the clock ($DCLK_X$) by at least T_{tsdx} or nominally 15 ns for that clock pulse to be recognized as the first clock pulse in the time slot. In actuality, the 2914 will allow FS_x to lag up to 80 ns the $DCLK_X$ rising edge and recognize it as the first clock pulse in a 2.048 MHz system.

Once FS_X has reached V_{IH} of about 2V, the D_X output will remain in the tri-state high-impedance mode for



NOTE:
All timing parameters referenced to V_{IH} and V_{IL} except t_{DON} and t_{OFF} which reference a high impedance state.

Figure 10. Variable Data Rate Timing for an Entire Time Slot

T_{don} or about 34 ns longer. It then comes out of tristate and will represent some data which is invalid until the valid data is available T_{DDX} or about 75 ns (100 ns worst case) after the clock rising edge. This means there is about 90 ns of invalid data after the tri-state mode. At this point there is valid data on the D_X highway that lasts for approximately one full clock cycle.

Since the D_X highway is tied directly to the D_r highway in digital loopback, the valid data above is now available to the receive channel with some propagation delay. The receiver is only interested in the data for about a 50 ns (110 ns worst case) window centered about the falling edge of the $DCLK_r$ clock which occurs about half a clock cycle from the FS_r rising edge. The window width is equal to the data set-up time. T_{dsr}, plus the clock fall time, T_f, plus the data hold time, T_{dhr}. Information at any other time on the D_r highway falls into the DON'T CARE category.

Retrofitting the 2913/14—Several switching/transmission systems have been designed using first-generation codecs which operate at data rates from 64 Kbps to 2.048 MBps. In addition, they may have been designed using the rising clock edges for both transmit and receive data.

Other aspects of these older designs could be relative skewing between the sync pulses (Data Enable) and the clock pulses in such a way that the sync pulse occurs after (Lags) the first clock pulse rising edge. All of these conditions can be easily handled using the variable data rate timing mode of the 2913/14 plus some simple external logic. By the addition of this logic, the 2913/14 becomes transparent to the older design thereby allowing an upgrade in performance while having no impact on backplane wiring or on system control hardware/software. In addition, many of the features of the 2913/14 may be incorporated, such as the test modes, which provide additional capabilities beyond those available in the original design and at a lower cost.

The circuit diagram in Figure 12 shows the maximum amount of additional random logic that could be necessary to make the 2913 or 2914 completely transparent at the linecard level (no impact on backplane wiring or timing). The inverter on $DCLK_R$ inverts all the receive clocks for each linecard. This inverter is only needed if (1) the transmit and receive clocks are inverted at the system/backplane level (as opposed to the linecard level) and (2) the previous design used only rising (or falling) edges to clock the transmit/receive data.

5

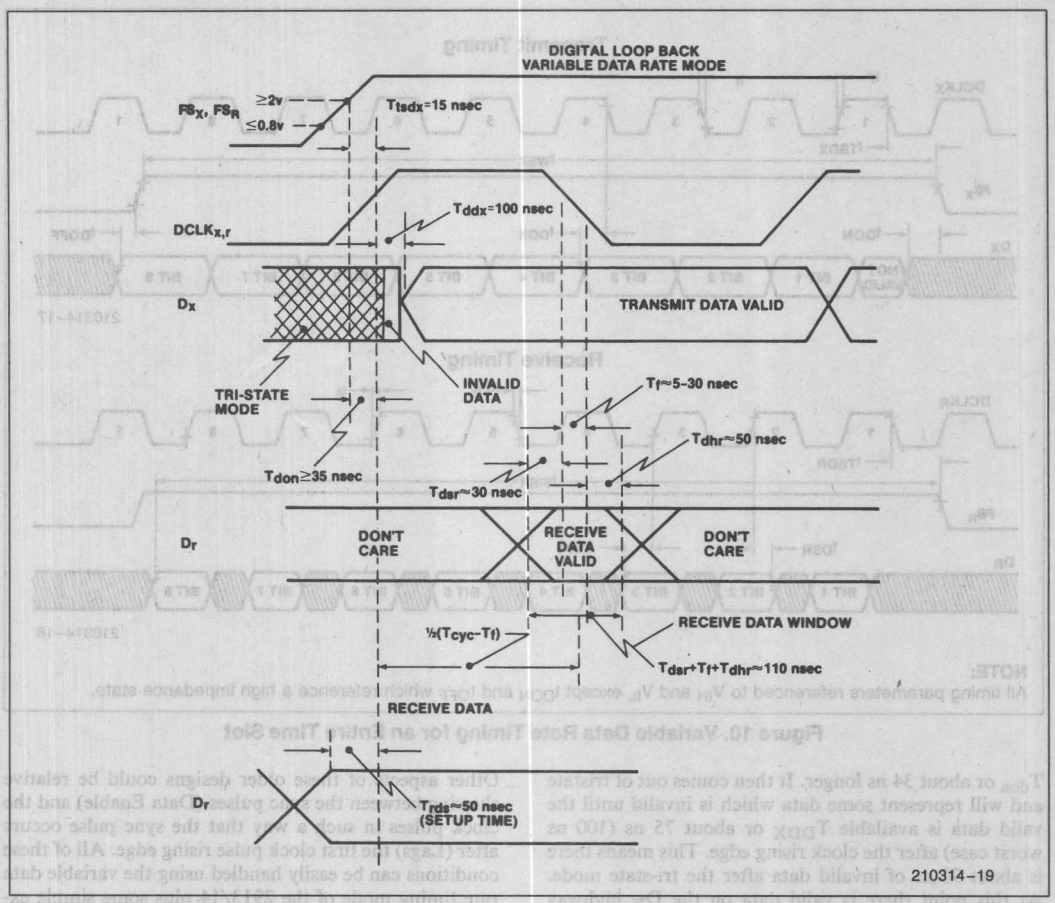


Figure 11. Waveform Timing Diagrams for the 2913/14

3.3 On-Line Test Modes

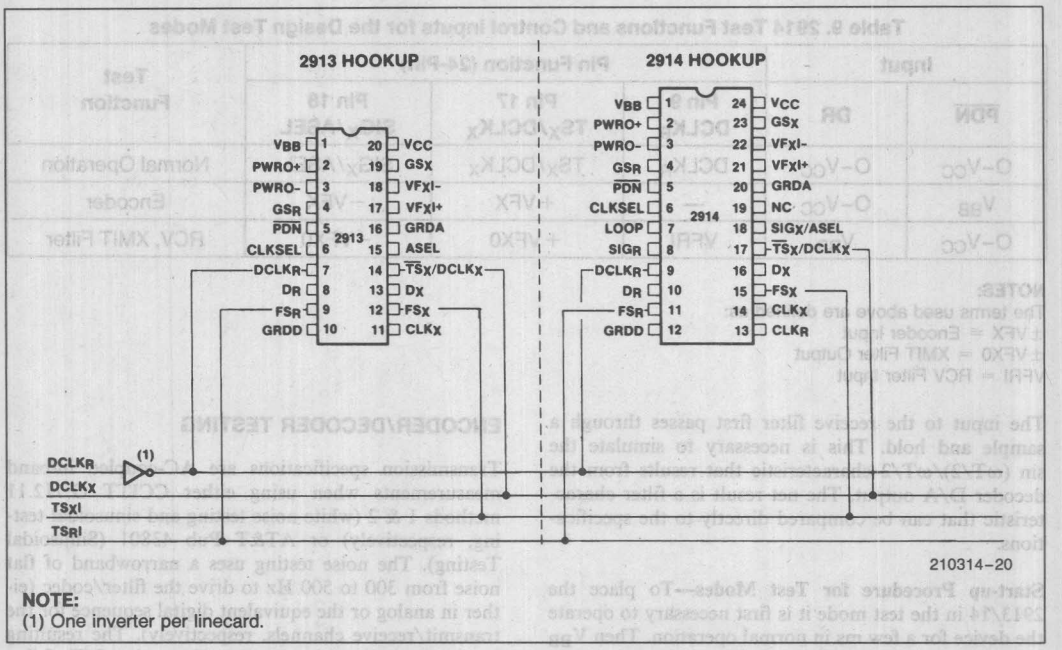
Two modes are available which permit maintenance checking of the linecard up to the SLIC/combochip interface, including the PCM highways and time slot interchanges. Tests include time slot-dependent error checking. The two test modes are called "redundancy testing" and "analog loopback." These test modes are described in detail in Section 4.3.

4.0 MULTIMODE TEST CAPABILITIES

The 2913/14 was designed with every phase of design, manufacturing, and operation taken into consideration. In particular, several test modes have been implemented within the device with essentially no increase in the package size or pin count. These test modes fall into three categories: design/prototype tests, manufacturing tests, and on-line operation tests; see Table 8.

4.1 Design/Prototype Testing

In the design of a linecard prototype or in the qualification of a device, it is often helpful to have direct access to the internal nodes at key points in the LSI system. Some manufacturers even dedicate pins specifically for this function. The Intel 2913/14 approach was to reduce cost by using multifunction pins and smaller packages to achieve this goal. Measurements through these multipurpose pins will typically yield full device capability against performance specifications, however *these measurements are not included in the device specifications.* This is done for two reasons: first, to save manufacturing cost by eliminating unnecessary tests and specifications, and, second, more cost effective manufacturing test techniques are available, as discussed in section 4.2.



NOTE:
(1) One inverter per linecard.

Figure 12. Circuit Diagram Showing Connections Needed to Retrofit the 2913/14 into Existing Variable Data Rate Systems

Table 8. Multimode Testing for Each Level from Design to On-Line Operation

- Design/Prototype Testing
  - — Direct access to transmit codec inputs
  - — Direct access to the receive filter input and the transmit filter differential outputs
- Manufacturing Tests
  - — Standard half channel tests for combined codec/filters
  - — Filter response half channel measurements
- Operation On-Line Tests
  - — Analog loopback for testing PCM and codec analog highways
  - — Redundancy checks with repeatable D_X outputs

Table 9 gives the input control pin values and the corresponding functions assigned to the key test pins on the 2914 for the design test modes.

Transmit Coded (Encoder)—The transmit filter can be bypassed by directly accessing the differential input of the transmit encoder with an analog differential drive signal. Table 9 shows the control pin voltages and the input pins for this test. This test mode permits DC testing of the encoder which is otherwise blocked by the AC coupling (low frequency reject filter) of the transmit filter.

Transmit and Receiver Filter—Table 9 shows the control values that permit access to the differential outputs of the transmit filter and the single-ended input to the receive filter. The voltage difference between the transmit filter outputs represents the filtered output that will be encoded. By driving VF_XI (single ended or differentially), the transmit filter response is obtained as a differential output. The final stage is the 60 Hz reject filter which is a switched capacitor filter sampled at an 8 KHz rate. When measured *digitally* (after the encoder), the filter characteristic is obtained directly: however, when measured in analog, a sin $(\omega T/2)/\omega T/2$ correction factor must be included.

5

Table 9. 2914 Test Functions and Control Inputs for the Design Test Modes

Input		Pin Function (24-Pin)			Test Function
$\overline{PDN}$	DR	Pin 9 $DCLK_R$	Pin 17 $TS_X/DCLK_X$	Pin 18 $SIG_X/ASEL$	
O–V_{CC}	O–V_{CC}	$DCLK_R$	$TS_X/DCLK_X$	$SIG_X/ASEL$	Normal Operation
V_{BB}	O–V_{CC}	—	+VFX	−VFX	Encoder
O–V_{CC}	V_{BB}	VFRI	+VFX0	−VFX0	RCV, XMIT Filter

NOTES:
The terms used above are defined as:
±VFX = Encoder Input
±VFX0 = XMIT Filter Output
VFRI = RCV Filter Input

The input to the receive filter first passes through a sample and hold. This is necessary to simulate the sin $(\omega T/2)/\omega T/2$ characteristic that results from the decoder D/A output. The net result is a filter characteristic that can be compared directly to the specifications.

Start-up Procedure for Test Modes—To place the 2913/14 in the test mode it is first necessary to operate the device for a few ms in normal operation. Then V_{BB} can be applied to the control pins to select the desired test access.

4.2 Production Testing

While it may be convenient for the designer to have access to both the filter and the codec inputs and outputs during the design or evaluation phase the final product will always use the filter and codec circuits together with all signals passing through both on the way to or from the PCM highways. It therefore makes sense to perform all manufacturing measurements with the device configured in its normal operating mode, i.e., all measurements should be complete filter/codec half channel measurements. This approach not only tests the combo as it will actually be used, but also saves time and money by eliminating separate measurements and correlation exercises to determine the full half channel performance.

Since the transmission specifications of S/D, gain tracking, and ICN all require measurements which are "in-band" or "filter independent," the codec functions can be easily tested using conventional half channel measurement equipment. The apparent difficulty arises in trying to fully measure the filter characteristics beyond the half sampling frequency of 4 KHz. In fact, this is not really a problem with today's computer-based testing plus an understanding of the sampled data process which is discussed under "Filter Testing".

ENCODER/DECODER TESTING

Transmission specifications are AC-coupled in-band measurements when using either CCITT G.712.11 methods 1 & 2 (white noise testing and sinusoidal testing, respectively) or AT&T Pub 43801 (Sinusoidal Testing). The noise testing uses a narrowband of flat noise from 300 to 500 Hz to drive the filter/codec (either in analog or the equivalent digital sequence for the transmit/receive channels, respectively). The resulting harmonic products are used to determine S/D. Likewise, gain tracking is also determined from this signal input. Sinusoidal testing uses a tone at 1.020 KHz for S/D measurements and gain tracking measurements. Idle channel noise measurements require the combined filter/codec since it has long been shown that separate measurements of filters and codecs are difficult to relate to the combined measurement (usually there is no specific relationship because of the non-linear properties of the encoder/decoder operations). Typically the frequency response of ICN measurements is primarily determined by the weighting filter (either C message or psophometric, which are both AC-coupled, bandpass type filters).

The conclusion is that combined filter/codec testing in no way limits the measurement of half channel transmission parameters of S/D, G.T., or ICN.

FILTER TESTING

Testing the filter response, of the transmit and receive channels presents two separate test situations which, in some ways, are mirror images of one another. With the transmit side, signals may be introduced at any frequency to test the filter response. At the output of the filter, the resulting signals are sampled at 8 KHz and digitized resulting in a sequence of PCM words representing the samples of filtered input signal. On the receive side, a digital PCM sequence of samples representing the driving signal is converted to an analog signal by the decoder and can be measured at the filter output in analog form.

Sampling Process—In both cases of testing the filter, the signal eventually is in a sampled form. Since the sampling rate is fixed at 8 KHz, all signals must be represented below 4 KHz (half the sampling frequency). This means that the PCM bit stream can only represent signals at frequencies below 4 KHz. If a signal above 4 KHz is sampled, those samples appear exactly as if the signal was at a frequency mirror imaged about 4 KHz. Two examples include signals at 5 KHz and 7 KHz which will result in samples that look like signals of 5–8 KHz = 3 KHz and 7–8 KHz = 1 KHz, respectively.

Conversely, the sampling process produces replicas (aliasing) of the sampled signal around multiples of the sampling frequency. Therefore, if two signals are introduced digitally representing 1 KHz and 2 KHz, there will also be frequency components located at 8 KHz = ±1 KHz and 8 KHz = ±2 KHz, and so on for all multiples of 8 KHz. Thus it is possible to generate frequencies at arbitrary values after sampling by controlling the frequency of each signal within the 4 KHz input band regardless of whether it is in analog or PCM.

When an analog signal is sampled, the frequency components generated are all of the same amplitude as the corresponding input spectral components. Therefore, on the transmit side, measurements made from the PCM data will have a throughput gain of unity except where components are superimposed (e.g., a 4 KHz input signal will have an alias component at 4 KHz which may double the amplitude at 4 KHz when the two components are combined).

When an analog signal is reconstructed from digital samples, it goes through a sample and hold stage which has the effect of imposing a weighting function on the resulting spectral components that is represented by

$$\text{Sinc}\left[\frac{\omega T}{2}\right] = \frac{\text{Sin}\left(\frac{\omega T}{2}\right)}{\frac{\omega T}{2}}$$

where ω is the actual spectral component frequency going into the filter, and T is the width of the hold pulse at the decoder output. For the 2913/14, the analog output is held the full sample period of 125 μs (1/8000 Hz) so that a frequency component at f_t will have a weighting of

$$W = \left(\frac{8000}{\pi f_t}\right) \text{Sin}\left[\frac{\pi f_t}{8000}\right]$$

Transmit Filter Test Approach—Two approaches can be used for half channel testing of the transmit filter characteristic: (1) input analog test frequencies and perform an FFT on the corresponding PCM samples that are generated to determine spectral frequencies and amplitudes at the codec output, or (2) use an "ideal" D/A converter on the PCM samples to convert the digital data back to analog so that the spectral amplitudes and frequencies can be determied using analog circuits such as spectrum analyzers or filter banks. In either case, the effects of sampling will be the same. Figure 13 shows two spectral diagrams of amplitude versus frequency. The top diagram represents the locations of nine test frequencies corresponding to the seven specified frequencies in the 2913/14 data sheet plus a component at 7 KHz and one at 10 KHz. The bottom figure shows the "equivalent" spectral component locations when carried in the PCM bit stream. As an example, frequency #8 is located at 7 KHz. The corresponding PCM frequency is seen in the lower figure at 1 KHz. Note also that the analog component at 9 KHz (see #8*) would also generate the 1 KHz component in the PCM data.

To test the filter, the desired test frequencies are introduced in analog to the filter input in such a way that there is no confusion as to where the resulting component will be after sampling (i.e., don't simultaneously put in 1 KHz and 7 KHz since both of these inputs result in a 1 KHz component in the PCM data). Then, using either technique (FFT or analog) mentioned above, measure the amplitude of the corresponding

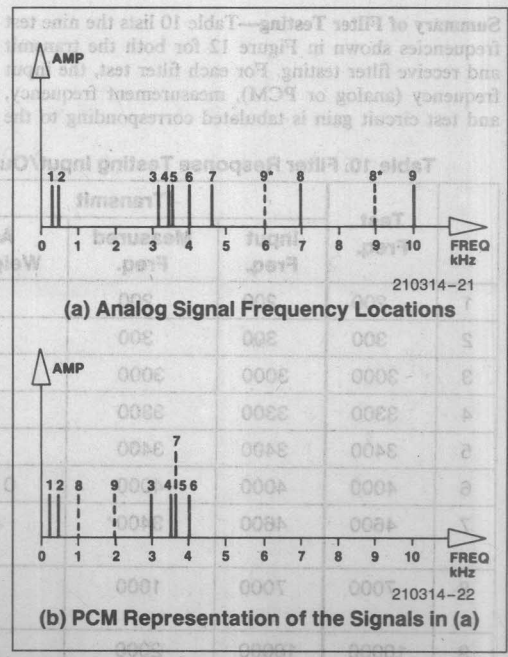


Figure 13. Spectral Properties of the Filter Test Frequencies in Analog and PCM

ple and hold output. At the filter output, the spectral component amplitudes will include the effect of the filter response *and* the weighting function measured at the actual test frequency. The receive filter includes a compensation network for the weighting function in its passband. Therefore, inside the passband (300 Hz to 3.4 KHz) the measured amplitudes should be compared directly to the data sheet specifications. Frequencies outside the passband must be compensated for the weighting function first to determine the true filter response.

Summary of Filter Testing—Table 10 lists the nine test frequencies shown in Figure 12 for both the transmit and receive filter testing. For each filter test, the input frequency (analog or PCM), measurement frequency, and test circuit gain is tabulated corresponding to the change network.

Redundancy Checking—A feature of the 2913/14 is that the same 8-bit PCM word can be put on the D_X highway in multiple time slots simply by holding the frame sync/data enable (FS_X) high and continuing to supply clock pulses (CLK_X or $DCLK_X$). If the data enable was held high for multiple time slots, each time slot would have identical data in it. By routing this data through the PCM highways, time slot interchanges, etc., and then correlating the data between time slots, it would be possible to detect time slot-dependent data errors. When this test mode is used, no other data will be generated for the transmit highway until the frame sync returns low for at least one full clock cycle.

Table 10. Filter Response Testing Input/Output Frequencies and Amplitude Gain Schedule

	Test Freq.	Transmit			Receive		
		Input Freq.	Measured Freq.	Amp Weighting	Input Freq.	Measured Freq.	Amp Weighting
1	200	200	200	1	200	200	1
2	300	300	300	1	300	300	1
3	3000	3000	3000	1	3000	3000	1
4	3300	3300	3300	1	3300	3300	1
5	3400	3400	3400	1	3400	3400	1
6	4000	4000	4000	0 to 2	4000	4000	0 to 2
7	4600	4600	3400	1	3400	4600	$\text{Sinc}\left[\frac{4600\,\pi}{8000}\right]$
8	7000	7000	1000	1	1000	7000	$\text{Sinc}\left[\frac{7000\,\pi}{8000}\right]$
9	10000	10000	2000	1	2000	10000	$\text{Sinc}\left[\frac{10000\,\pi}{8000}\right]$

Analog Loopback—The 2914 (2913 does not have this feature) has the capability to be remotely programmed to disconnect the outside telephone lines and tie the transmit input directly to the receive output to effect analog loopback within the combo chip. This is accomplished by setting the LOOP input to V_{CC} (TTL high). The result is to disconnect VF_XI+ and VF_XI- from the external circuitry and to connect internally PWRO+ to VF_XI+, GS_r to PWRO−, and VF_XI- to GS_X (see Figure 14).

With this test set up, the entire PCM and analog transmission path up to the SLIC can be tested remotely by assigning a PCM word to a time slot that is read by the combo being tested. This data is converted to analog and passed out of the receive channel. It is taken as input by the transmit channel where it is filtered and redigitized (encoded) back to PCM. The PCM word can now be put on the transmit highway and sent back to the remote test facility. By comparing the PCM data (individually or as a series of codes) the health of that particular connection can be verified.

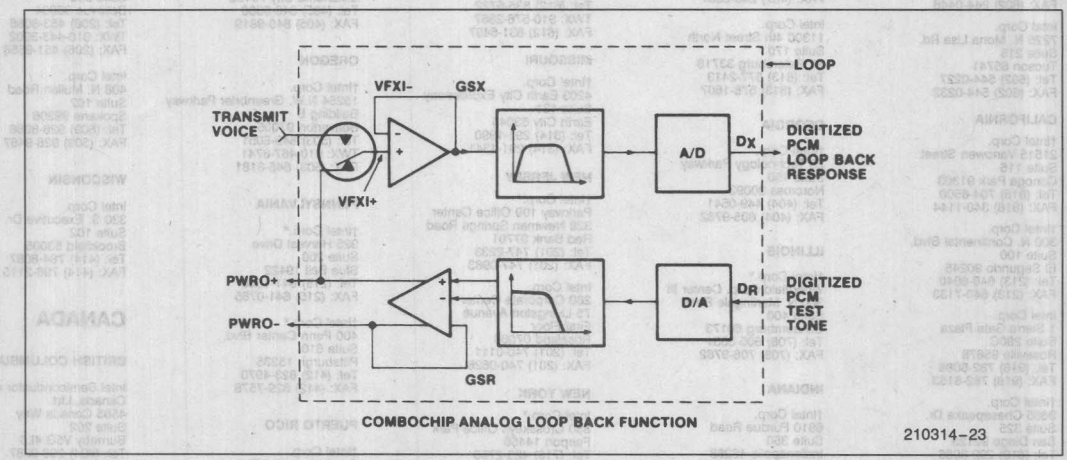


Figure 14. Simplified Block Diagram of 2914 Combochip in the Analog Loopback Configuration

5

DOMESTIC SALES OFFICES

ALABAMA

Intel Corp.
5015 Bradford Dr., #2
Huntsville 35805
Tel: (205) 830-4010
FAX: (205) 837-2640

ARIZONA

†Intel Corp.
410 North 44th Street
Suite 500
Phoenix 85008
Tel: (602) 231-0386
FAX: (602) 244-0446

Intel Corp.
7225 N. Mona Lisa Rd.
Suite 215
Tucson 85741
Tel: (602) 544-0227
FAX: (602) 544-0232

CALIFORNIA

†Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500
FAX: (818) 340-1144

†Intel Corp.
300 N. Continental Blvd.
Suite 100
El Segundo 90245
Tel: (213) 640-6040
FAX: (213) 640-7133

Intel Corp.
1 Sierra Gate Plaza
Suite 280C
Roseville 95678
Tel: (916) 782-8086
FAX: (916) 782-8153

†Intel Corp.
9665 Chesapeake Dr.
Suite 325
San Diego 92123
Tel: (619) 292-8086
FAX: (619) 292-0628

†Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114
FAX: (714) 541-9157

†Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
2nd Floor
Santa Clara 95051
Tel: (408) 986-8086
TWX: 910-338-0255
FAX: (408) 727-2620

COLORADO

Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (719) 594-6622
FAX: (303) 594-0720

†Intel Corp.*
600 S. Cherry St.
Suite 700
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289
FAX: (303) 322-8670

CONNECTICUT

†Intel Corp.
301 Lee Farm Corporate Park
83 Wooster Heights Rd.
Danbury 06810
Tel: (203) 748-3130
FAX: (203) 794-0339

FLORIDA

†Intel Corp.
800 Fairway Drive
Suite 160
Deerfield Beach 33441
Tel: (305) 421-0506
FAX: (305) 421-2444

†Intel Corp.
5850 T.G. Lee Blvd.
Suite 340
Orlando 32822
Tel: (407) 240-8000
FAX: (407) 240-8097

Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33716
Tel: (813) 577-2413
FAX: (813) 578-1607

GEORGIA

†Intel Corp.
20 Technology Parkway
Suite 150
Norcross 30092
Tel: (404) 449-0541
FAX: (404) 605-9762

ILLINOIS

†Intel Corp.*
Woodfield Corp. Center III
300 N. Martingale Road
Suite 400
Schaumburg 60173
Tel: (708) 605-8031
FAX: (708) 706-9762

INDIANA

†Intel Corp.
8910 Purdue Road
Suite 350
Indianapolis 46268
Tel: (317) 875-0623
FAX: (317) 875-8938

IOWA

Intel Corp.
1930 St. Andrews Drive N.E.
2nd Floor
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

†Intel Corp.
10985 Cody St.
Suite 140
Overland Park 66210
Tel: (913) 345-2727
FAX: (913) 345-2076

MARYLAND

†Intel Corp.*
10010 Junction Dr.
Suite 200
Annapolis Junction 20701
Tel: (301) 206-2860
FAX: (301) 206-3677
(301) 206-3678

MASSACHUSETTS

†Intel Corp.*
Westford Corp. Center
3 Carlisle Road
2nd Floor
Westford 01886
Tel: (508) 692-0960
TWX: 710-343-6333
FAX: (508) 692-7867

MICHIGAN

†Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48322
Tel: (313) 851-8096
FAX: (313) 851-8770

MINNESOTA

†Intel Corp.
3500 W. 80th St.
Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867
FAX: (612) 831-6497

MISSOURI

†Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990
FAX: (314) 291-4341

NEW JERSEY

†Intel Corp.*
Parkway 109 Office Center
328 Newman Springs Road
Red Bank 07701
Tel: (201) 747-2233
FAX: (201) 747-0983

Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111
FAX: (201) 740-0626

NEW YORK

Intel Corp.*
850 Crosskeys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391
FAX: (716) 223-2561

†Intel Corp.*
2950 Express Dr., South
Suite 130
Islandia 11722
Tel: (516) 231-3300
TWX: 510-227-6236
FAX: (516) 348-7939

†Intel Corp.
Westage Business Center
Bldg. 300, Route 9
Fishkill 12524
Tel: (914) 897-3860
FAX: (914) 897-3125

NORTH CAROLINA

†Intel Corp.
5800 Executive Center Dr.
Suite 105
Charlotte 28212
Tel: (704) 568-8966
FAX: (704) 535-2236

†Intel Corp.
5540 Centerview Dr.
Suite 215
Raleigh 27606
Tel: (919) 851-9537
FAX: (919) 851-8974

OHIO

†Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528
FAX: (513) 890-8658

†Intel Corp.*
25700 Science Park Dr.
Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298
FAX: (804) 282-0673

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (405) 848-8086
FAX: (405) 840-9819

OREGON

†Intel Corp.
15254 N.W. Greenbrier Parkway
Building B
Beaverton 97005
Tel: (503) 645-8051
TWX: 910-467-8741
FAX: (503) 645-8181

PENNSYLVANIA

†Intel Corp.*
925 Harvest Drive
Suite 200
Blue Bell 19422
Tel: (215) 641-1000
FAX: (215) 641-0785

†Intel Corp.*
400 Penn Center Blvd.
Suite 610
Pittsburgh 15235
Tel: (412) 823-4970
FAX: (412) 829-7578

PUERTO RICO

†Intel Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

Intel Corp.
8911 Capital of Texas Hwy.
Austin 78759
Tel: (512) 794-8086
FAX: (512) 338-9335

†Intel Corp.*
12000 Ford Road
Suite 400
Dallas 75248
Tel: (214) 241-8087
FAX: (214) 484-1180

†Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490
FAX: (713) 988-3660

UTAH

†Intel Corp.
428 East 6400 South
Suite 104
Murray 84107
Tel: (801) 263-8051
FAX: (801) 268-1457

VIRGINIA

†Intel Corp.
1504 Santa Rosa Road
Suite 108
Richmond 23288
Tel: (804) 282-5668
FAX: (216) 464-2270

WASHINGTON

†Intel Corp.
155 108th Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002
FAX: (206) 451-9556

Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086
FAX: (509) 928-9467

WISCONSIN

Intel Corp.
330 S. Executive Dr.
Suite 102
Brookfield 53005
Tel: (414) 784-8087
FAX: (414) 796-2115

CANADA

BRITISH COLUMBIA

Intel Semiconductor of
Canada, Ltd.
4585 Canada Way
Suite 202
Burnaby V5G 4L6
Tel: (604) 298-0387
FAX: (604) 298-8234

ONTARIO

†Intel Semiconductor of
Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
FAX: (613) 820-5936

†Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
FAX: (416) 675-2438

QUEBEC

†Intel Semiconductor of
Canada, Ltd.
1 Rue Holiday
Suite 115
Tour East
Pt. Claire H9R 5N3
Tel: (514) 694-9130
FAX: 514-694-0064

†Sales and Service Office
*Field Application Location